社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2004－ＳＬＤＭ－113　(13)
2004／1／23

クロック木の配線長を考慮したクロックスケジュール法の改良

山崎　創†　高橋　篤司†

† 東京工業大学大学院理工学研究科集積システム専攻
〒 152-8552 東京都目黒区大岡山 2-12-1
E-mail: †{h-yama,atushi}@lab.ss.titech.ac.jp

あらまし　準同期式回路において，位置的に近いレジスタに同一のクロックタイミングを割り当てること（クラスタ分割）により高速なクロック周期を実現しながら，クロック配線長を抑えることができるクロックスケジュール手法が提案された．この手法はレイアウト情報から初期分割を得て，修正を繰り返す．本研究では、実際にクリティカルサイクル上にあるレジスタに着目し，効率的にクリティカルサイクルを解消するようにクラスタ修正を行う手法を提案した．提案手法を実際の回路データに使用したところ，安定してクロック周期を高速化することができ，クロック配線長も既存手法と比較して改善が見られた．

キーワード　準同期回路，クロック木，クロックスケジュール，クラスタ，クリティカルサイクル

Improvement of Clock Scheduling Method in Consideration of Clock Tree Length

Hajime YAMAZAKI† and Atsushi TAKAHASHI†

† Department of Communications and Integrated Systems, Tokyo Institute of Technology
Ookayama 2-12-1, Meguro-ku, Tokyo, 152-8552 Japan
E-mail: †{h-yama,atushi}@lab.ss.titech.ac.jp

Abstract　In order to achieve a shorter clock period by a clock tree with less wire length and less power consumption, a clustering based clock scheduling algorithm was proposed. In the algorithm, first registers are partitioned into clusters by their locations, and then clusters are modified to improve the clock period by changing the cluster to which a critical register belongs. However, in the modification of clusters, excessive modifications are carried out because critcal registers which break cycle are not identified well. In this paper, we propose a method that reduces the number of modification by identifying critical register. Critical registers are identified by extracting non trivial strongly connected components of critical constraint graph. In experiments, the validity of the enhancement is confirmed.

Key words　semi-synchronous circuit, clock-tree, clock schedule, cluster, critical cycle

1. はじめに

近年，各レジスタへ必ずしも同時刻にクロックを分配しない設計方式，準同期式設計が提案され，その実用化へ向けた検討が進められている．完全同期式回路ではレジスタ間最大遅延よりクロック周期を短くできないが準同期式回路では多くの場合においてレジスタ間最大遅延よりも短いクロック周期が実現できる．また，完全同期式設計と準同期式設計を比較した場合，同じクロック周期を達成するクロック木の消費電力は準同期設計のほうが小さくすることが可能である [3], [5].

短いクロック周期を実現するには各レジスタにクロックが到達するタイミング (クロックスケージュール) を適切に設定し，そのタイミングでクロックが到達するようクロック木を構築せねばならない．

しかし任意のクロックスケジュールを達成するクロック木の構築は可能ではあるが，クロック木の構築を考慮せずにクロックスケジュールを決定するとそれを実現するためのクロック配線が非常に困難となり，配線長も長くなるため実用的とはいえない [6]．ランダムにクロックスケジュールを設定した場合は，位置的に近いレジスタに対して同じようなタイミングを与えた場合に対して，経験的にはより長いクロック配線が必要とされる [2].

そこで配線長や消費電力が小さい短いクロック周期を実現するクロック木を構成するために，位置的に近いレジスタに同じタイミングを与えるアルゴリズム CBCS [5] や，クロック配線長について改良を加えた修正版 [9] が提案されている．

アルゴリズム CBCS では，同一クラスタ内のレジスタは同じクロックスケジュールを与えることを前提に，クラスタ内のレジスタ数およびレジスタの分布範囲を制限し，レジスタ集合をクラスタに分割する．CBCS はまずチップ領域を菱形領域に分割し，各領域内のレジスタ集合をそれぞれ初期クラスタとし，各レジスタの所属クラスタを変更していく．

クラスタ分割をせずに各レジスタに自由にクロックを与えることができたなら，準同期回路として限界の最小クロック周期が達成できる．各クラスタに，自由にクロックを与えることができるとき，クラスタ分割が決定されるとその分割での最小クロック周期が決定する．現在のクラスタ分割が，準同期の限界クロック周期まで到達していないならば，クラスタ分割を修正することで，クロック周期を短縮できる可能性がある．

CBCS では，所属しているクラスタを変更することでクロック周期を短縮できる可能性があるレジスタを探索し，そのレジスタの所属クラスタを変更する．しかし，その探索手法が不完全であり，高速化を阻害しているレジスタのみを発見することはできなかったので，余分なクラスタ修正も行なわれていた．余分なクラスタ修正が行なわれると，クロック配線長が増大してしまう．

そこで本論文ではアルゴリズム CBCS を基本として，スケジューリングを行なうアルゴリズムを改良し，余分な修正は行わずに高速化を阻害しているレジスタのみに対して効率的にクラスタ修正ができるようにした手法を提案する．

提案手法を 888，7672，7052 個のレジスタからなる回路に対して適用し，既存手法と比較した．初期分割を変化させたとき，既存手法よりも，得られた最終分割時のクロック周期のばらつきが少なく，配線長では優れているクラスタ分割を得ることができ提案手法の効果が確認された．

2. 準同期式回路

完全同期式設計では，各レジスタへ同じタイミングでクロックを入力している．しかし，準同期式設計では各レジスタに入力するクロックの周期は同じであるが，同時に入力することは前提としない．

レジスタ v にクロックを供給するタイミングを $s(v)$ とする．準同期回路において，レジスタ u からレジスタ v へ信号が伝搬するとき，クロックタイミング $s(u), s(v)$ が満さなければならない制約条件は以下のように表すことができる [1], [7].

No-Double-Clocking (Hold) Constraint

$$s(v) - s(u) \leqq d_{min}(u,v)$$

No-Zero-Clocking (Setup) Constraint

$$s(u) - s(v) \leqq T - d_{max}(u,v)$$

ここで，$d_{\min}(u,v)$ ($d_{\max}(u,v)$) はレジスタ u からレジスタ v への信号伝搬の最小（最大）遅延を，T はクロック周期を示す．この制約がすべての信号伝搬のあるレジスタ対に対して満されるとき，回路は正常に動作するとする．

制約グラフ $G_T(V,E)$ をレジスタを点，クロック周期 T のときの制約を枝としたグラフとする．制約 $s(v) - s(u) \leqq w_T(u,v)$ に対し u から v への重み $w_T(u,v)$ の枝を加える．図 2 は図 1 に示す回路の制約グラフである．

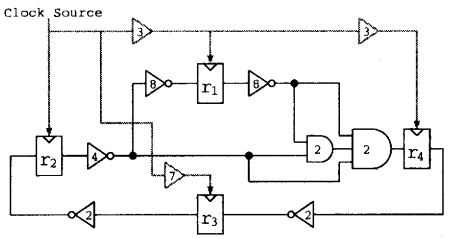


図 1　準同期式回路

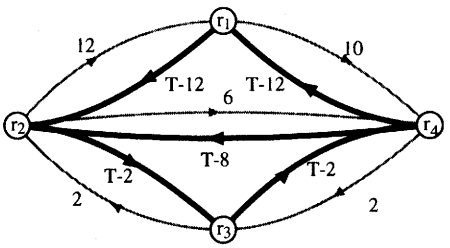


図 2　制約グラフ $G_T(V,E)$

レジスタ間の最大・最小遅延が与えられ，各レジスタのクロックが任意に設定できるとき，準同期式回路がクロック周期 T で動作するか否かは $G_T(V,E)$ 中に負閉路があるかどうかで判定できる．$G_T(V,E)$ が負閉路を持たずに零閉路が存在している時，クロック周期 T は最小クロック周期を達成している．$G_T(V,E)$ が重み和が正の閉路のみで構成されていた場合さらにクロック周期を小さくすることが可能である．

$G_T(V,E)$ が負閉路を持たないとき回路を正常に動作させるクロックスケジュールが存在する．すべての制約を満たすクロックスケジュールを許容スケジュールと呼ぶ．図 1 の回路を最小クロック周期 $T = 9$ で動作させるための許容スケジュールを図 3 に示す．ただし，T における許容スケジュールは一般に一意に定まるわけではない．

T における許容スケジュールを s とする．各レジスタに対し，$s(v)$ を含むクロックタイミングの範囲 $r_s(v)$ を定める．各レジスタ v のクロックタイミングを $r_s(v)$ から任意に選択して得られるクロックスケジュールが，許容スケジュールであるとき，$r_s(v)$ をクロック周期 T におけるスケジュール s に対する v のスケジュール可能範囲と呼ぶ．

クロック周期 T で許容スケジュール s が与えられたとき，$G_T(V,E)$ 上の枝 (u,v) に対応する制約 $s(v) - s(u) \leqq w_T(u,v)$

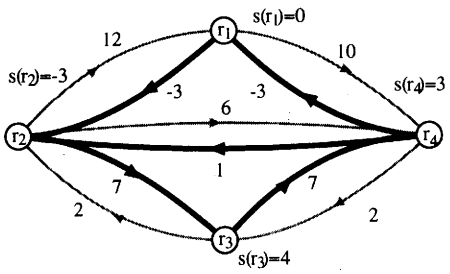


図 3　制約グラフ $G_9(V,E)$ とクロックタイミング

は満たされる．このとき，$s(v) - s(u) = w_T(u,v)$ ならば枝 (u,v) をクリティカルと呼ぶ．

制約グラフ $G_T(V,E)$ 上にクリティカル枝からなるサイクル (パス) が存在する時，そのサイクル (パス) をクリティカルサイクル (パス) と呼ぶ．枝がクリティカルであるか否かは許容スケジュールによって異なるが，クリティカルサイクルは許容スケジュールによらず定まる．クリティカルサイクル上の枝重み和はクリティカル枝の定義より零となっていることがわかる．

本稿では，レジスタ集合をいくつかのクラスタに分割するが，各クラスタ内のレジスタのクロックタイミングは等しいと仮定する．レジスタ集合がクラスタ $C_1, C_2, \ldots, C_m$ に分割されたとする．制約グラフ $G_T(V,E)$ から $C_i (1 \leqq i \leqq m)$ に含まれる点を一点に縮退して得られるグラフをクラスタ制約グラフ $G_T^C(V^C, E^C)$ とする．クラスタ内のレジスタのクロックタイミングが等しいという制約の元での最小クロック周期 $T(G^C)$ および $T(G^C)$ における許容クロックスケジュール s^C は，$G_T^C(V^C, E^C)$ を用いて得ることができる．このとき，s^C における各レジスタのスケジュール可能範囲は $G_T(V,E)$ および s^C を用いて求める．クラスタ C のスケジュール可能範囲 $r_s(C)$ はクラスタ内に含まれるレジスタのスケジュール可能範囲の共通部分とする．

クラスタ制約グラフにおいても，制約グラフと同様にクリティカルサイクルは定義される．C_i に属するレジスタ u から C_j に属するレジスタ v への制約に対応するクラスタ制約グラフ上の枝 (C_i, C_j) が，クラスタ制約グラフのクリティカルサイクル上の枝であるとき，u, v をクリティカル点と呼ぶ．

3. CBCS の概要

提案手法はアルゴリズム CBCS [5], [9] を改良した手法であるため，まず CBCS を説明する．

CBCS では，レイアウトを考慮しながらレジスタ集合を一つのバッファで駆動されるいくつかのクラスタに分割する．このとき，クロック源から各クラスタのクロックバッファへの遅延時間は自由に設定できると仮定する．また，各クラスタ内のレジスタのクロックタイミングは同一であるとする．駆動バッファからレジスタへの距離が大きくなるとその間の配線による遅延の影響を大きく受けるようになり，クラスタ内のレジスタのクロックタイミングを同時であるとみなせなくなる．そこで，クラスタ内のレジスタのチップ上での位置を囲む最小マンハッタン円 (正菱形) の半径 (以降，クラスタ半径と呼ぶ) に制限を加える．クラスタの最大半径は駆動バッファから各レジスタまでの配線による遅延を無視できるように設定する．このため，クラスタ内のレジスタのスケジュールは同一とみなすことができる．

CBCS では，チップ領域を菱形領域に分割し各菱形領域内のレジスタ集合をそれぞれ初期クラスタとし，クロック周期を短縮するために，制約グラフ上でクリティカルサイクルを解消できるよう，レジスタの所属クラスタの変更を繰り返すことで最終的なクラスタ分割を得る．あるクラスタのレジスタを隣接するクラスタへ所属させたとき所属先のクラスタの半径が最大半径を越えないように初期クラスタの半径は最大半径の 1/3 以下とする．

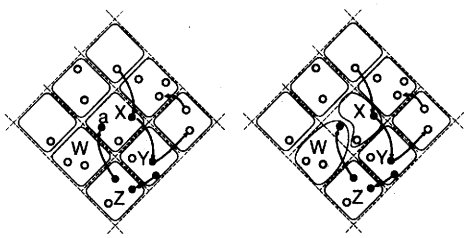
クラスタ数を多くすればクロック周期は短くできるが，クロック源から各バッファまでの配線総長が長くなると考えられるため実用的でない．したがって，バッファが駆動可能なレジスタ数なども考慮の上，クラスタ数はできるだけ少ない数に抑える．

4. 提案手法

4.1 既存手法からの改良点

アルゴリズム CBCS では，クロック周期を高速化するために，クリティカル点の所属クラスタを変更してクリティカルサイクルを解消する．

図 4 はクラスタ分割にしたがって制約グラフを縮退した例であり，丸はレジスタ，レジスタを囲んでいる閉曲線がクラスタ，枝はクリティカル枝を表している．また，黒丸はクリティカル点を表している．図 4(a) では，X, Y, Z の 3 クラスタから構成されるクリティカルサイクルができているが，X に所属しているレジスタ a の所属クラスタを W に変更することで，図 4(b) では，クラスタ制約グラフのクリティカルサイクルが解消できていることがわかる．



(a)　(b)

図 4　クリティカルサイクルの解消 (a) 解消前 (b) 解消後

しかし，既存手法で使われている修正アルゴリズムでは，クラスタ制約グラフからクリティカル点のみの抽出はできなかった．既存手法では，得られているクラスタ分割で最小クロック周期と各レジスタのクロックスケジュールを求め，その最小クロック周期より少し小さいクロック周期でスケジューリングを行ない，もともとのクロックスケジュールと同じスケジュールを与えることのできないレジスタを制約グラフより抽出していた．

例えば，図 4(a) のクラスタ制約グラフでは図 5 に示す黒丸が修正候補集合として選択される．このように，抽出されないクリティカル点が存在したり，クリティカル点でないレジスタも抽出してしまう．そのため既存手法では，確実にクリティカルサイクルを解消するために，抽出したレジスタすべてに対してクラスタ修正を試みていた．

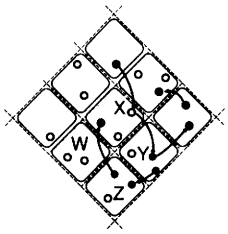


図 5　既存手法による修正候補の抽出

有向グラフの強連結成分とは，任意の 2 頂点 u,v に対して u から v へ至るパスと，u から v へと至るパスが存在する部分グラフである．有向グラフの強連結成分への分割は線形時間で可能である [10]．

提案手法ではクラスタ制約グラフからクリティカル枝以外を削除したクラスタクリティカルグラフを作成する．任意のクリティカルサイクルは，クラスタクリティカルグラフのある 2 点以上の強連結成分に含まれる．2 点以上の強連結成分の点は，クリティカルサイクル上の点である．したがって，クリティカル点が属するクラスタは，クラスタクリティカルグラフの 2 点以上の強連結成分に含まれる点に対応する．そこで，そのグラフの 2 点以上の強連結成分を抜き出し，クラスタ制約グラフのクリティカルサイクル上のクラスタとする．次に，クリティカルサイクル上の枝の重みを決定しているレジスタを特定することで，クリティカル点を抽出することを可能とした．

図 4 の例において提案手法を適用すると，クラスタ制約グラフから，図 4(a) に示す黒丸が修正候補集合として選択される．

このように，修正候補のレジスタを一つだけクラスタ修正すればクリティカルサイクルを解消することが可能となり，余分なクラスタ修正を行なわずに済むようになった．

4.2　提案手法の概要

提案手法ではチップ領域を菱形領域に分割する．各菱形領域は最大 8 菱形領域と隣接する．各菱形領域には初期クラスタと高々一つの新規クラスタを対応させる．レジスタ v の位置する菱形領域を $h(v)$ とする．$h(v)$ を含む，$h(v)$ に隣接する菱形領域を $H(v)$ とする．クラスタ分割が与えられたとき，レジスタ v に関する修正候補は，v が所属するクラスタから，そのクラスタを除く $H(v)$ に対応し，レジスタ v とスケジュール可能範囲に共通部分が存在するクラスタへの修正である．

入力　レジスタ間最大遅延 $d_{\max}(u,v)$
レジスタ間最小遅延 $d_{\min}(u,v)$
レジスタ配置位置
最大半径 R
出力　クラスタ分割 $C_1,C_2,\ldots,C_m$，各クラスタのクロックタイミング

（1）　チップ領域を菱形領域に分割し，各菱形領域内のレジスタ集合をそれぞれ領域に対応する初期クラスタとする．ただし，菱形領域の半径は最大半径 R の 1/3 倍以下，かつ，各初期クラスタ内のレジスタを一つのバッファで駆動できるように設定する．

（2）　得られたクラスタ分割における最小クロック周期 $T(G^C)$ とその周期における各レジスタ v のクロックスケジュール $s^C(v)$，各クラスタのスケジュール可能範囲を求める．

（3）　クラスタ制約グラフからクリティカル点を求め S とする．

（4）　S に含まれるすべてのレジスタに関する修正候補のうち，修正によるクラスタ内配線長の増加が最小である修正を行なう．レジスタの修正が行なわれれば，(2) へ戻る．

（5）　$h(v)$ で新規クラスタが生成されていない集合 S 中のレジスタのうちで，最もスケジュール可能範囲の広いレジスタ v に対して，新規クラスタを生成しレジスタ v を新規クラスタへ移動させる．新規クラスタが生成されれば，(2) へ戻る

（6）　新規クラスタに属するレジスタ v に対し，次に初期分割時のクラスタに属さないレジスタ v に対し，レジスタ v のスケジュール可能範囲と初期分割時のクラスタのスケジュール可能範囲に共通部分が存在するとき，レジスタ v を初期分割時のクラスタへ移動する．

（7）　クラスタ分割，最小クロック周期，クロックスケジュールを出力し，終了．

5.　実　　験

回路	面積 [μm^2]	レジスタ数	レジスタ間最大遅延 [ps]	フラット時クロック周期 [ps]
a	728 × 710	888	11569	8323
b	1950 × 1909	7672	11808	9552
c	1958 × 1958	7052	8354	6256

表 1　使用した回路

提案手法の効果を確認するために表 1 に示される回路を用いて実験を行った．表にあるフラット時クロック周期とはクラスタ分割をせずに，準同期化したもので，準同期式回路として限界のクロック周期である．すべての回路において配線遅延が無視できる長さから最大半径 R は 300[μm] とした．また，比較している既存手法は [9] で提案されている配線長重視の CBCS である．

なおクラスタ内配線においては，アルゴリズム CRBST [4] が使用されている．CRBST ではソースからシンクまでの配線経路長の最大許容値をパラメータで制御できる．配線抵抗による遅延が無視できる配線長にするため，最大クラスタ半径を各クラスタ内配線における最大許容値として設定する．

クラスタ間配線においては，スケジュール可能範囲を持つクラスタを扱うために，修正版 SC [2] を用いる．修正版 SC は

バッファの挿入に対してマージグラフを用いる．短い配線でマージすることができ，消費電力が少なくてすむクラスタのペアが選択され，ボトムアップで再帰的にクラスタをマージしていく．制約として，駆動バッファからの最大配線長とバッファの最大駆動容量が与えられている．

5.1 初期分割の変化に対する実験

提案手法と既存手法を，表1に示される回路に適用した．提案手法をC言語で実装し，Pentium4 3.06GHz において実行したところ回路 (a) で約4分，回路 (b)，(c) では約18分である．提案手法の実行時間は既存手法の約2倍となっている．

回路 (a) については，初期分割を垂直・水平方向に初期分割の半径の1/20ずつずらし計400通り，回路 (b)(c) については1/5ずつ計25通り適用した．

図6，図7，図8にクロック周期とクラスタ内配線長の値を示す．proposedは提案手法，existingは既存手法であり，X軸はクロック周期 (ps)，Y軸はクラスタ内配線長 (mm) である．

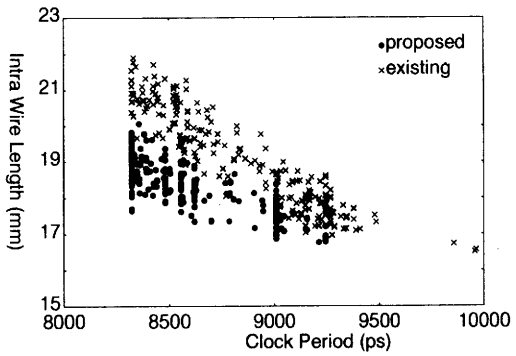


図6 クロック周期-クラスタ内総配線長分布 回路 (a)

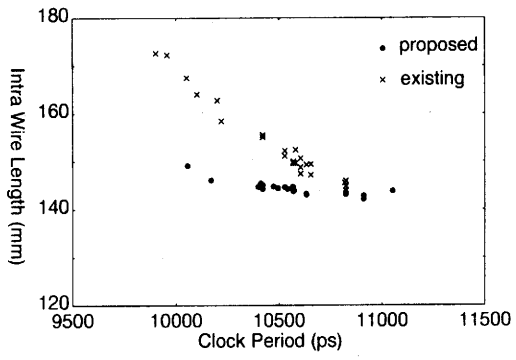


図7 クロック周期-クラスタ内総配線長分布 回路 (b)

5.2 クラスタ間配線

既存手法，提案手法を適用し得られた，クロック周期が最小の分割のなかで，クラスタ内配線長が最小のクラスタ分割に対し，クラスタ間配線を行った．回路 (a)(b)(c) に対する結果がそれぞれ，表2，表3，表4である．ここで“フラット”はクラスタ分割を行わず，そのまま配線アルゴリズム (修正版SC) を適用した結果を示している．“初期”，“既存手法　最終”，“提案手法　最終”はそれぞれ初期分割，既存手法による最終分割，提案手法による最終分割に対しクロック木を構成した結果である．( ) 内は分割クラスタ数を表す．配線長の内訳は，クラスタ内部の配線の総長 (intra) とクラスタ間配線長 (inter) の計2つである．

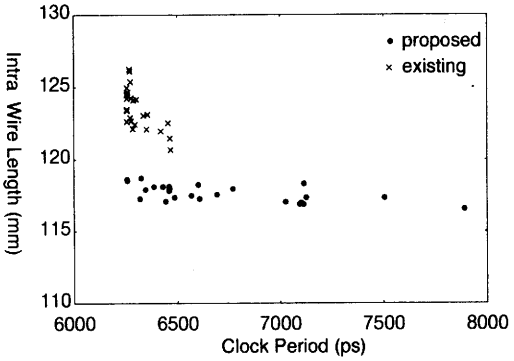


図8 クロック周期-クラスタ内総配線長分布 回路 (c)

	クロック周期 [ps] (%)	クロック配線長 [μm] (%)	(intra, inter) [μm]	#buf
フラット	8323 (100)	39857 (130)	(— , —)	207
初期 (37)	10154 (122)	23540 (77)	(15410, 8130)	56
既存手法 最終 (43)	8323 (100)	30579 (100)	(20370, 10209)	84
提案手法 最終 (46)	8323 (100)	28253 (92)	(17644, 10609)	94

表2 回路 a(レジスタ間最大遅延 11569) 最小クロック周期，配線長

	クロック周期 [ps] (%)	クロック配線長 [μm] (%)	(intra, inter) [μm]	#buf
フラット	9552 (96)	326989 (138)	(— , —)	1559
初期 (220)	11307 (114)	193696 (82)	(140370, 53326)	374
既存手法 最終 (256)	9904 (100)	236124 (100)	(172640, 63484)	396
提案手法 最終 (239)	10056 (102)	207399 (89)	(149117, 58282)	355

表3 回路 b(レジスタ間最大遅延 11808) 最小クロック周期，配線長

	クロック周期 [ps] (%)	クロック配線長 [μm] (%)	(intra, inter) [μm]	#buf
初期 (212)	7829 (123)	165796 (95)	(116881, 48915)	287
既存手法 最終 (220)	6341 (100)	174159 (100)	(122153, 52006)	315
提案手法 最終 (241)	6256 (99)	174652 (100)	(118672, 55980)	320

表4 回路 c(レジスタ間最大遅延 8354) 最小クロック周期，配線長

6. 考　　察

6.1 回路 (a) について

回路 (a) でクラスタ内配線とクラスタ間配線を行った結果では，提案手法では既存手法とともに，準同期式回路としての最小クロック周期に到達するとともに，クロック配線長では既存手法と比較して 8%改善した．このときクラスタ修正をしたレジスタ数は 94 レジスタであり，最終分割時のクラスタ数は 46 である．表 6 の，初期分割を 400 通り変化させた結果からも，ほぼすべての場合で既存手法よりも短いクロック配線長を実現でき，提案手法の有効性が確認できた．

得られた最小クロック周期の分布を図 9(既存手法) と図 10(提案手法) に示す．X 軸はクロック周期 (ps)，Y 軸はそのクロック周期が現れた回数である．既存手法ではクロック周期に分布に広がりがあるが，提案手法では特定のクロック周期に固まっている．特にフラットでの最小クロック周期である 8323(ps) に固まっており，高速化の安定性という面でも既存手法からの改善がみられる．

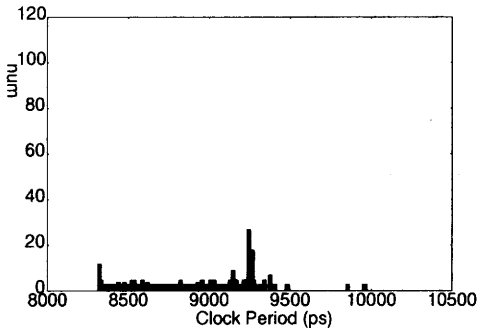


図 9　既存手法クロック周期分布

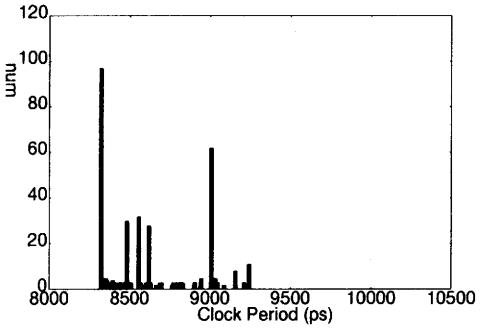


図 10　提案手法クロック周期分布

6.2 回路 (b) について

図 7 より，同程度のクロック周期では既存手法より短いクラスタ内配線長を実現できたことがわかる．しかし，最小クロック周期では既存手法のほうが，短いクロック周期を達成しており，さらなる高速化は今後の課題である．また，クラスタ内配線とクラスタ間配線を行った結果 (表 3) では，提案手法では既存手法にクロック周期では 2%ほど悪化したが，クロック配線長では既存手法と比較して 11%改善した．

6.3 回路 (c) について

図 8 より，同程度のクロック周期では既存手法より短いクラスタ内配線長を実現できたことがわかる．さらに，最小クロック周期でも提案手法では，準同期としての限界であるフラット時と同じクロック周期も達成できた．また，クラスタ内配線とクラスタ間配線を行った結果 (表 4) では，提案手法では既存手法ではほぼ同等の結果である．

7. 結論と今後の課題

本稿において効率的にクロック周期の高速化が可能なクロックスケジュール法を提案した．このアルゴリズムによって，準同期回路において高速なクロック周期を短い配線長で実現することが可能となった．また今後の課題は，クリティカルサイクルの構造を考慮する効率的な初期分割の選択および，クラスタ修正法を提案し，クロック配線長はおさえつつも，クロック周期をさらに短縮する高速アルゴリズムを提案することである．

謝　　辞

実験において用いたデータを提供していただいた，松下電器産業株式会社安井卓也氏，黒川圭一氏に感謝する．

文　　献

[1] J. P. Fishburn. Clock skew optimization. *IEEE Trans. on Computers*, Vol. 39, No. 7, pp. 945–951, 1990.

[2] K. Inoue, W. Takahashi, A. Takahashi, and Y. Kajitani. Schedule-clock-tree routing for semi-synchronous circuits. *IEICE Transactions on Fundamentals*, Vol. E82-A, No. 11, pp. 2431–2439, 1999.

[3] K. Kurokawa, T. Yasui, M. Toyonaga, and A. Takahashi. A practical clock tree synthesis for semi-synchronous circuits. *IEICE Transactions on Fundamentals*, Vol. E84-A, No. 11, pp. 2705–2713, 2001.

[4] H. Mitsubayashi, A. Takahashi, and Y. Kajitani. Cost-radius balanced spanning/Steiner trees. *IEICE Transactions on Fundamentals*, Vol. E80-A, No. 4, pp. 689–694, 1997.

[5] M. Saitoh, M. Azuma, and A. Takahashi. Clustering based fast clock scheduling for light clock-tree. In *Proc. Design Automation and Test in Europe Conference and Exhibition (DATE)*, pp. 240–244, 2001.

[6] A. Takahashi, K. Inoue, and Y. Kajitani. Clock-tree routing realizing a clock-schedule for semi-synchronous circuits. In *Proc. IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pp. 260–265, 1997.

[7] A. Takahashi and Y. Kajitani. Performance and reliability driven clock scheduling of sequential logic circuits. In *Proc. Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 37–42, 1997.

[8] T. Yoda and A. Takahashi. Clock schedule design for minimum realization cost. *IEICE Transactions on Fundamentals*, Vol. E83-A, No. 12, pp. 2552–2557, 2000.

[9] 山崎創, 高橋篤司. クロック木の配線長を考慮したクラスタ修正によるクロックスケジュール法. VLD (VLD2002-35), Vol.102, No.164, 電子情報通信学会技術報告書, 2002.

[10] 石畑清. アルゴリズムとデータ構造, 岩波ソフトウェア科学, 第 3 巻. 岩波書店, 1989.