システムＬＳＩ設計技術　93－6
（1999. 11. 27）

複数クロックを考慮したタイミング制約生成手法

松永 多苗子　田宮 豊　松永 裕介

(株) 富士通研究所
〒211-8588 川崎市中原区上小田中 4-1-1
E-mail: kakuda@flab.fujitsu.co.jp

あらまし

本稿では，同期式順序回路を対象としたスタティックタイミング解析において，複数クロックにより規定されるタイミング条件を扱う枠組みについて提案する．複数の異なるクロックによって駆動される回路では，各FFに対する到着時刻，要求時刻はパスによって異なり，一意に決めることができない．そこで，一括して扱うことのできるタイミング制約集合 CCS(Compatible Constraint Set) を複数用意するアプローチを提案する．

キーワード　スタティックタイミング解析

An Approach To Generate Timing Constraints Considering Multiple Clocks

Taeko MATSUNAGA, Yutaka TAMIYA and Yusuke MATSUNAGA

FUJITSU LABORATORIES LTD.
4-1-1 Kamikodanaka, Nakahara-ku, Kawasaki, 211-8588, JAPAN
E-mail: kakuda@flab.fujitsu.co.jp

Abstract

In this paper, we propose an approach to treat timing constraints for synchronous sequential circuits with multiple clocks. Compatible constraint sets, called CCS, are introduced and a heuristic to generate a set of CCS are shown.

key words　static timing analysis

1 はじめに

本稿では，ハイパフォーマンスを目指した設計において，周期や位相が異なる複数のクロックが存在する同期式順序回路のタイミングを解析する問題について取り上げる．

同期式順序回路のタイミングは，FF 間のパスの遅延時間と，FF を駆動するクロックとの関係によって規定される．例えば，図 1 において，FF1 から出力されたデータが，FF2 において正しくが取り込まれるためには，FF2 にクロックエッジが到着する時刻の前後でデータが安定している必要がある．FF1 のクロックピン CK から，FF2 の入力ピン D までの遅延時間を Td，クロック周期をτ，セットアップ時間，ホールド時間を Tsu, Th とすると，この条件は Td に対する以下のような式で表される．

$$Td + Tsu \leq \tau, Td \geq Th$$

このように,FF 間のパスには，最大遅延時間や最小遅延時間等の，遅延時間制約が存在していて,Td とそれらの制約とを比較することによってタイミングの検証が行える．

遅延時間 Td は，FF の CK ピンに到達時刻を設定し，入力から出力に向かってトポロジカルオーダで遅延時間を計算していくことによって求められる．そして，FF の D ピンに設定した要求時刻と比較することによって，制約が満たされているかどうかが判断できる．この場合，FF 間の各パスごとに遅延計算を行うのではなく，回路中の複数のパスを一括して扱うことができる．

ただし，クロックが複数存在した場合には，到達時刻や要求時刻を一つに決められない場合がでてくる．Td に対する時間制約は，送り側と受取り側の FF のクロックエッジの差によって決められる (図 2)．例えば，最大遅延時間に関する制約を考えた場合，単一クロックの場合には，図 (1) のように一クロック後になるが，2 相クロックであれば,$\tau/2$ になる．さらに周期が異なる場合には，クロックエッジの差の中でもっとも近接するものが制約になる．

また，回路中のパスの中には，1 クロックではなく 2 クロックで到達すればよいパスもあり，タイミング解析においてそのようなパスをマルチサイクルパスとして指定することが可能である．このような場合にも，時間制約は異なったものになる．

このように，回路中に複数のクロックが存在すると，パスごとに時間制約が異なったものになる可能性がある．例えば，図 3 のような回路を考える．FF3 には FF1 からのパスと FF2 からのパスが存在し，FF4 にも FF1 から のパスと FF2 からのパスが存在するとする．

この場合，どこからきたパスかによって，終点側の FF で取り込まれる時刻が異なってくる．FF1 から FF3 へのパスの時間制約は 10，FF1 から FF4 への時間制約は 5 であるから，FF1 の到達時刻を 0 とすると，FF3, FF4 の要求時刻は，それぞれ 10, 5 となる．次に FF2 からのパスについて考えると FF3 へのパスは 5, FF4 へのパスは 10 である．FF2 の到着時刻を 5 とすると，FF3, FF4 の要求時刻は 10, 15 になる．すると，FF4 では，FF1 からのパスに関しては要求時刻 10, FF2 からのパスに関しては 15 となり，すべてのパスの時間制約を一括して解析することはできなくなる．

そこで，複数のクロックやマルチサイクルパスの指定が存在する回路から導かれるタイミング制約を扱うために，一括して扱うことのできるタイミング制約集合 CCS(Compatible Constraint Set) という概念を導入し，これらを複数用意することでタイミング制約を扱う枠組みを提案する．以下では，まず CCS を用いたタイミング解析の全体像について簡単に述べた上で，CCS の定義，および,CCS を生成するアルゴリズムについて説明する．

2 複数のクロックを考慮したタイミング解析

図 4 に，両立可能なタイミング制約 CCS を用いたタイミング解析の流れを示す．

まず回路構造を解析して，パスが存在する FF 対を抽出し，FF 間の時間制約を生成してタイミング制約グラフを生成する．その上で，すべてのタイミング制約をカバーする CCS 集合を生成し，CCS ごとに遅延計算を行ってすべてのタイミング制約に対するチェックを行う．

CCS の生成は，最初に一回行えば，与えられるタイミング制約が変らない限りは，同一である．ただし,CCS ごとの遅延計算は，回路構造が変われば再計算する必

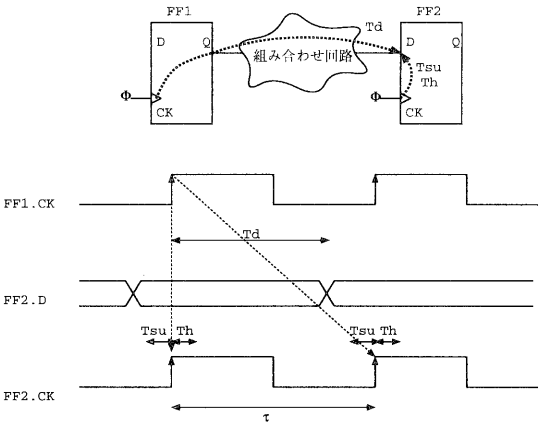
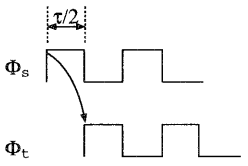
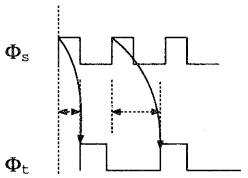


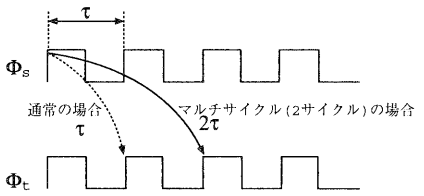
図 1: 同期式順序回路のタイミング



2相クロックの場合



周期や位相が異なる場合



マルチサイクル(2サイクル)の場合

図 2: クロックが単一でない場合

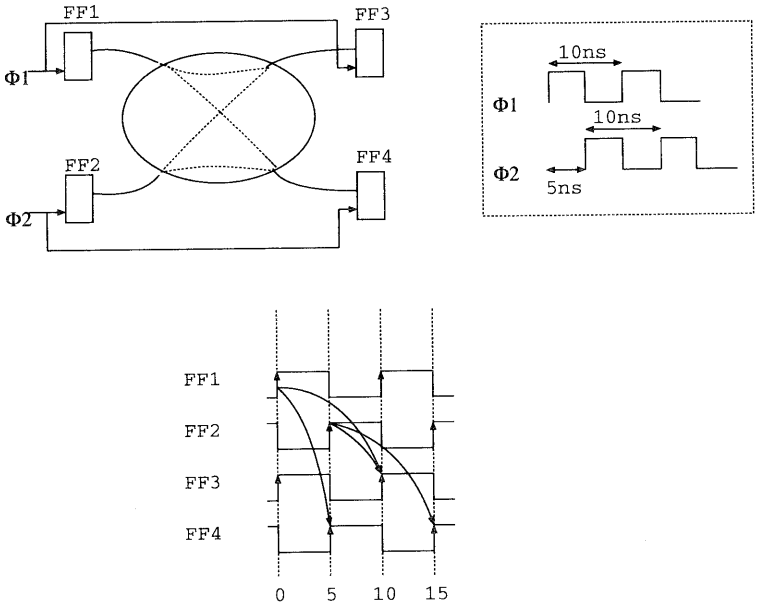


図 3: パスによって時間制約が異なる例

要がある.したがって,論理合成などのように,遅延計算結果に基づいて回路の構造を変更する,ということを繰り返す場合などを考えると,CCS の個数は,できるだけ少ない方が望ましい.

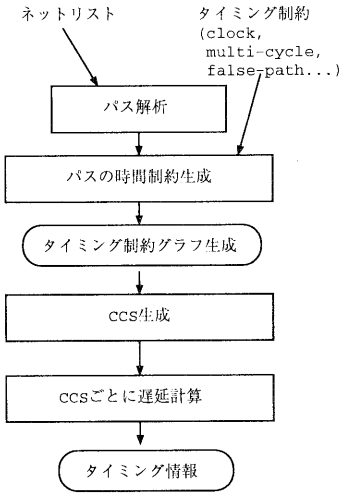


図 4: CCS を用いたタイミング解析の流れ

以下では,FF 対とパスの時間制約はすでに計算されているものとして,CCS と CCS 生成アルゴリズムについて述べる.

3 両立可能なタイミング制約集合 CCS

3.1 CCS の定義

定義 1 (タイミング制約グラフ) :

2 部グラフ $G=(V_s,V_t,E)$ を考える.ここで,

- $v_s \in V_s$: 始点 FF_s に対応
- $v_t \in V_t$: 終点 FF_t に対応
- $e_{st} \in E$: FF_s,FF_t 間にパスが存在することを表す

ものとする.さらに,各エッジには,FF 間の時間制約 T_{diff} に対応する重み w が以下のように定義されているものとする.

$$w(e_{st}) = T_{diff}(s,t)$$

$$w(e_{ts}) = -w(e_{st})$$

このようなグラフ G を,タイミング制約グラフと呼ぶものとする.

例 1 (タイミング制約グラフ) :

図 3 のような回路構造,およびクロック定義からは,図 5 のようなタイミング制約グラフが生成される.(ここで,グラフ中には,v_s から v_t へのエッジの重みが記されている.)

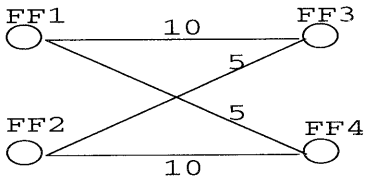


図 5: タイミング制約グラフ

タイミング制約グラフ上で,以下のように CCS を定義する.

定義 2 (CCS) :

両立可能なタイミング制約 *CCS(Compatible Constraint Set)* とは,タイミング制約グラフ G のサブグラフ $H=(V_s',V_t',E')$ で,以下の条件を満たすものである.

1. E' は,G 上の V_s', V_t' 間のすべてのエッジを含む
2. H に含まれるすべての閉路について,エッジの重みの合計が 0 になる.

例 2 (CCS) :

図 6 の *(a),(b)* は図 5 の 2 部グラフに対する *CCS* である.*(c),(b)* は,それぞれ,条件 1, 2 を満たさないため,*CCS* ではない.

CCS H が連結で,かつ,H には含まれない G 中のノードを加えると CCS の条件を満たさなくなってしまう場合,H は 連結な maximal CCS であるという.

3.2 連結な maximal CCS を生成するアルゴリズム

アルゴリズム 1 :

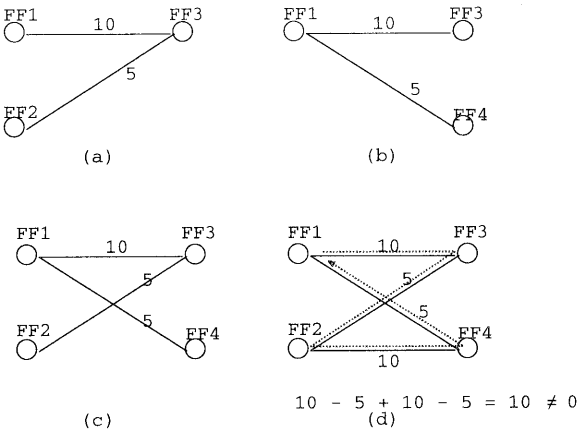


図 6: CCS になる場合とならない場合の例

1. タイミング制約グラフ G の各ノードの t を未設定としておく.
2. 1つのノード $v_0 \in V_s \cup V_t$ を選び, $t(v_0) = 0$ とする. 空の CCS を生成して, CCS に v_0 を加える.
3. v_0 を出発点としてグラフを辿り, 辿れるノードがなくなるまで以下を繰り返す:

   - ノード v_i からノード v_j に辿った場合の $value$ を, $t(v_i) + w(e_{ij})$ で計算する.

     case1: $t(v_j)$ が未設定の場合, $value$ を設定して v_j を CCS に加え, v_j の先のノードを辿る.

     case2: $t(v_j)$ がすでに設定されている場合

     case2-1: その値が $value$ に等しいならば, OK

     case2-2: その値が $value$ に等しくない場合, このエッジが加わらないように, ノード v_i あるいは v_j を CCS からはずす.

     どちらの場合も, v_j から先は辿らない.

このアルゴリズムでは, 一つのノードを始点としてそこからエッジを辿りながらノードを CCS に加えて時刻を設定していき, 時刻に矛盾がおきたらノードを取り除く, ということを繰り返す.

生成される CCS は, ノードの探索の仕方と, ノードの値に矛盾が起きたときに取り除くノードの選び方に依存する.

4 タイミング制約集合の生成

すべてのタイミング制約をカバーする CCS 集合を求める問題を, 以下のように定義する.

定義 3 (CCS 生成問題) : CCS 生成問題とは, G のすべてのエッジがどこかのサブグラフ H に含まれるように, H の集合を生成することである.

CCS 生成問題を厳密に解く方法としては, すべての maximal な CCS を生成して, タイミング制約グラフのすべてのエッジに対して最小被覆問題を解くことが考えられる. ただし, 実回路においては, タイミング制約グラフのノード数, エッジ数はそれぞれ, 数万, 数十万のオーダーになることを考え, ここでは, 下記のような方針に基づくヒューリスティックを考案した.

1. 連結な maximal CCS を生成する.
2. 連結な CCS を生成するごとに, そこに含まれるエッジにマークをつけ, すべてのエッジにマークがつくまで, くりかえす.
3. マージできる CCS をマージして, CCS 数を減らす.

4.1 すべてのエッジをカバーする連結なCCS集合の生成

基本的には，アルゴリズム1をタイミング制約グラフ中の各ノードに対して順に実行して，CCSに含まれたエッジにマークをつけ，すべてのエッジにマークがつくまで繰り返す．

一旦マークのついたエッジを他のCCSでカバーする必要はない．ただし，マークのついていないエッジを辿って生成したノード集合に，マークのついたエッジの両端ノードが含まれる場合には，そのエッジも必然的にCCSに含まれるので，条件のチェックを行う必要がある．

そこで，アルゴリズム1を以下のように修正して，連結CCSの生成に用いる．

アルゴリズム 2 ：アルゴリズム *1* の *3.* において時刻のチェックを行う際，

- *vi* から *vj* へのエッジが *unmark* であった場合は，アルゴリズム *1* と同様．
- エッジにマークがついていた場合は，
  - *vt* が未到達ノードであるならば，何もしない
  - *vt* に時刻が既に設定されている場合，時刻に矛盾がなければ何もしない．矛盾があるならば，マークがないノードと同様の処理を行う．

これを用いて，すべてのエッジをカバーするCCS集合は，以下のように求める．

- すべてのエッジをunmarkに設定する
- unmarkのエッジがなくなるまで，以下を繰り返す
  - アルゴリズム2を実行して，連結なCCSを生成する
  - CCSに含まれるエッジにマークをつける

連結なCCSを生成する際に，必ず1つ以上のエッジはマークされるので，プログラムの終了は保証される．

4.2 CCSのマージ

2つのCCS $H_1 = \{V'_{s1}, V'_{t1}, E'_1\}$, $H_2 = \{V'_{s2}, V'_{t2}, E'_2\}$ を考える．このとき，$\forall v_{s1(2)} \in V'_{s1(2)}, \forall v_{t2(1)} \in V'_{t2(1)}$ に対して，両ノードを端点とするエッジが存在しないならば，

$$H_3 = \{V'_{s1} \cup V'_{s2}, V'_{t1} \cup V'_{t2}, E'_1 \cup E'_2\}$$

もCCSである．すなわち，両CCSに交わりがなく，マージしてもエッジが増えない場合，CCSの条件は満たされたままなので，1つのCCSとしてマージすることができる．

そこで，アルゴリズム2を実行後に，上記条件をチェックし，マージできるものをまとめてCCSの個数を減らす．

5 考察

上記アルゴリズムによってどのようなCCSが生成されるかは，ノードを選択する順，エッジの探索のしかたに依存する．これらの選び方が，生成されるCCSの数にどのように影響を与えるかについては，今後アルゴリズムの実装，実験を通して評価していく必要がある．

現実的な回路においてCCSの数がどのくらいになるかについては，回路の性質やタイミング制約の種類に依存するため，一概には言えない．ただし，予備調査によると，10万セル規模の回路(FF数は数万個)数個に対して，セットアップ条件のためのCCSの数は数個～10数個程度であった．これより，クロックが複数する場合でも，それらはある程度グループ化されており，異なるクロックが複雑に絡み合うようなケースはあまり多くないように思われる．ただし，数個であっても，繰り返し遅延計算が必要になる場合を考えると，できるだけ少なくしておく必要があると思われる．

6 おわりに

本稿では，複数のクロックが存在する回路のタイミング解析手法として，両立可能な制約集合CCSを用意し，それぞれについて遅延計算を行うアプローチを示した．また，CCSを生成するアルゴリズムについての提

案を行った．今後，本アルゴリズムの実装，評価を行うとともに，この枠組みのなかでフォールスパスを扱う方式についての検討を行っていく予定である．

7 謝辞

本研究を進めるにあたって，貴重な議論や情報の提供をしてくださった (株) 富士通研究所の金澤裕治氏に感謝致します．

参考文献

[1] DESIGN COMPILER リファレンスマニュアル: CONSTRAINTS AND TIMING, VERSION 1999.05 , Synopsys