Vol.2017-SLDM-180 No.5
2017/5/10

隣接するブロック間のみに配線をもつ FPGA に対する配置配線手法

丸岡大浩[†1]　藤田昌宏[†2]

概要：FPGA の集積度が増すにつれ，FPGA の低コスト化と高速化が進んだ一方でタイミングクロージャ問題が大きな課題となっている．そこでタイミングクロージャ問題が原理的に生じないようにするため，隣接する論理ブロック間のみに配線を持つ FPGA のモデルについて検討した．本論文では配置配線の際の制約を整数線形計画問題（ILP）として定式化することにより配置配線を行う手法を提案し，実際にベンチマーク回路を用いてモデルの FPGA に対する配置配線が可能か否かを確かめた．

Mapping and routing method for the FPGA which has wires between only neighbor blocks

TOMOHIRO MARUOKA[†1]　MASAHIRO FUJITA[†2]

1. はじめに

FPGA （Field Programmable Gate Array）は任意の回路を実装できるプログラマブル素子の一種である．半導体プロセスの微細化により FPGA の集積度が増すにつれ FPGA の低コスト化と高速化が進んでいる．しかし，近年 FPGA の高集積密度化が進む一方で，配線由来の遅延はロジックセルや記憶セル由来の遅延よりも大きくなっており，論理合成時に見積もった遅延が配置配線後の遅延と一致しないといった問題も生じている．これはタイミングクロージャ問題と呼ばれ，一度生じると解消されるまで配置配線を繰り返したり，場合によっては論理回路も変更したりする必要がある．そのため 1 回あたりの配置配線にかかる時間が大きい大規模回路では，配置配線を繰り返すことで設計期間が増大するといった重大な問題を引き起こしている．

この問題を解消する方法の一つとして，タイミングクロージャ問題を解消する FPGA を用いるというものが挙げられる．こういった FPGA には様々なものが考えられるが，この論文では隣接する論理ブロック間のみに配線を持ち LUT の出力は必ずレジスタに値を保存することで配線長をおよそ均一にした FPGA を対象とする．本研究では 3.1 項で後述する参考研究をもとに，この FPGA への配置配線を整数線形計画問題（ILP）として定式化し ILP ソルバーを用いて解くことにより行う手法を提案する．その後実際にベンチマーク回路を用いて配線領域を制限した FPGA へ配置配線することができるか否かを確かめる．ILP に基づく配置配線手法を大規模回路にそのまま適用することはできないため，回路を分割して適用する手法を提案し，評価する．

†1 東京大学大学院工学系研究科電気系工学専攻
Department of Electronic Engineering, The University of Tokyo
†2 東京大学大規模集積システム設計教育研究センター
VLSI Design and Education Center, The University of Tokyo

2. 配線領域を制限した FPGA

2.1 配線領域を制限した FPGA

配線領域を制限した FPGA として最も単純なものは，従来の FPGA からスイッチブロックやコネクションブロックなどを取り除いた隣接する論理ブロック間のみに配線を持つものが挙げられる．しかし配線領域を制限したことで論理ブロックをデータ移動の中継地点としても用いないといけない場合もあるため，実際の FPGA のように論理ブロック内に LUT が 1 つだけしか存在しないとすると，データ移動と LUT の配置が競合し配置配線が不可能になることが考えられる．そのためブロック内に複数の LUT を有し動作させることで，データ移動と LUT の配置の競合を解消し，配線長を制限したままタイミングクロージャ問題を解消することができると考えた．次節以降ではこれらの特徴を持つ FPGA のモデルを考え，その構造について述べる．

2.2 隣接するブロック間のみに配線を持つ FPGA

この節で説明する FPGA は，図 1 のような複数の LUT を含む LUT Block をメッシュ状に並べ，隣接する LUT Block 間のみに配線を有した FPGA である．隣接する LUT Block との間にしか配線資源が存在しないため，隣接する LUT 間の配線長はおよそ等しくなり回路の最大・最小遅延の予測が容易になるため，タイミングクロージャ問題を回避できると考えられる．また配線長が比較的短いため 1 サイクルあたりにかかる時間も短くなりより高周波数での動作が期待できると考える．一方で配線要素が LUT Block 間に限られるため，離れた LUT Block に配置された LUT 間のデータ移動には LUT Block を通過する必要がある．そのため離れた LUT 間のデータ移動には数サイクルかかることも起こり得る．

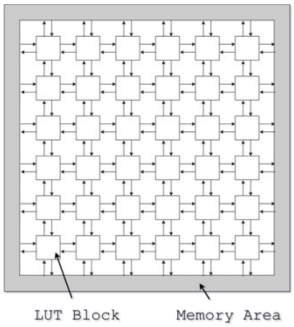

図 1　隣接するブロック間のみに配線を持つ FPGA

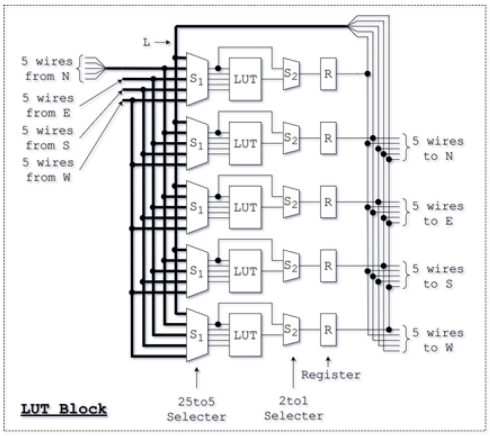

図 2　LUT Block の構造

次に LUT Block の詳細について説明する．図 2 は 5 個の 5 入力 LUT を含み，隣接する LUT Block との間には双方向 5 本ずつの計 10 本の配線を有する LUT Block を示している．LUT の入力方向についているセレクタは上の LUT Block（N），右の LUT Block（E），下の LUT Block（S），左の LUT Block（W）からそれぞれ来た 5 本ずつの配線と，自身の LUT Block の 1 サイクル前の出力データを伝播する 5 本の配線を加えた計 25 本の配線から LUT の入力数である 5 本を選択し出力する．セレクタ S1 から出力されたデータは LUT に入力されるが，その内の 1 つは LUT を経ずに LUT の出力とともにセレクタ S2 に入力される．これは LUT Block がデータ移動の中継地点となる際に，LUT をバッファとして用いてこれを行うと LUT Block の面積が増大してしまうがゆえに設けてあるバイパス配線である．セレクタ S2 の出力は必ずレジスタに保存される．そのためこのアーキテクチャの配線長はあるレジスタから別のレジスタまでの距離に等しく，その距離は推測しやすいと考えられる．そして LUT Block に含まれる LUT の数だけあるレジスタの出力はそれぞれ N,E,S,W の LUT Block へと送られる．

また実際の FPGA の I/O ブロックに相当するものとして，図 1 のような FPGA の外側を囲むメモリ領域を有するものとする．そのため配置配線したい回路の入力データはこのメモリ領域から LUT Block に伝播し，出力データは LUT Block からメモリ領域に伝播される必要がある．メモリ領域と FPGA の外周の LUT Block とは配線により接続されているが，この配線数は LUT Block 間のそれとは異なり，メモリ領域から LUT Block への 1 本と逆方向の 1 本の計 2 本とする．そのため同時に 1 つの LUT Block が保持できる回路の入力データと出力データはそれぞれ 1 つずつとなる．以下，この隣接するブロック間のみに配線を持つ FPGA を単位 FPGA と呼ぶ．

3. ILP モデル

3.1 参考研究

単位 FPGA への整数線形計画問題を用いた配置配線手法を提案するにあたり，伝播遅延を持つ命令の配置を分散レジスタモデルにいくつかの制約を加え拡張した ILP ベース手法を提案した論文[1][2]を参考とした．この論文では C プログラムをデータフローグラフ（DFG）に変換し，命令とデータ移動が満たすべき制約をコンパイラによって生成した後，ILP ソルバーを用いて制約を満たす解の組を探索することで，図 3 のように DFG をメッシュ状のプロセッサアレイ上にマッピングすることが可能となったと結論づけている．

この論文では 1 つのプロセッサが異なるいくつかの命令を実行可能である．そこで本研究で提案した 5 つの LUT を持つ LUT Block を，5 種類の論理関数を実行可能なプロセッサであるかのように見なすことで，この論文の提案手法を応用して制約式を考え回路の配置配線を行った．

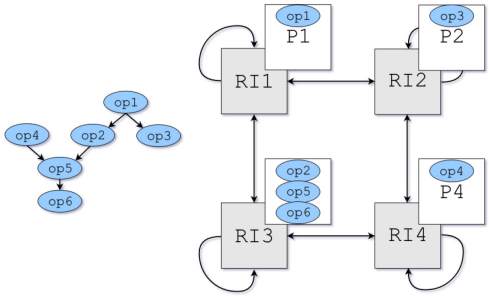

図 3　DFG のメッシュ状プロセッサアレイへの配置

3.2 ILP

整数線形計画問題（ILP）とは変数を整数に限定した線形計画（LP）問題を解く手法を指す．LP とは線形な等式および不等式を制約とし，線形関数を目的関数としたときに目的関数を最大化若しくは最小化する変数の値を求める手法である．

3.3 提案する ILP モデル

このモデルでは，まず配置配線したい回路を n 入力以下の LUT のみから構成される回路に変換する．その後回路中の LUT をノードに，LUT と LUT をつなぐ配線をエッジとみなした DFG として扱う．順序回路は FF の出力と入力をそれぞれ回路の入力と出力として見なすことで，組み合わせ回路として扱う．まずノードの配置を表すために 2 値変

数$X(op,t,p)$を定義する．op はノードを，t は op が配置されるサイクルを，p は op が配置される LUT Block を指す．つまり$X(op,t,p)=1$という式は回路中の op に相当する LUT が時間 t に LUT Block p の LUT に配置されるということを示している．

次に DFG のエッジを表すために別の 2 値変数$D(i,t,p,d)$を定義する．i はエッジを，t は i のデータ移動が行われるサイクルを，p と d は i のデータ移動が LUT Block p から d 方向への配線で行われることを示す．つまり$D(i,t,p,d)=1$という式はデータ i の移動が時間 t に回路中の LUT Block p から d 方向の LUT Block へ向かう配線を用いて行われるということを示している．方向 d については，単位 FPGA では上下左右の LUT Block を示す 4 種類と自身に回帰する 1 種類の計 5 種類存在する．これ以降文中では LUT Block p から d 方向への配線を$p(d)$と表す．

このように DFG におけるノードとエッジに関する 2 値変数を導入したが，ILP ソルバーを用いて配置配線を行うためにはいくつかの制約が必要となる．

まずノードに関する制約式を定義する．以下では制約式のうち基本的なものについて述べる．詳細については文献[1][3]を参照されたい．全てのノードは何れかのサイクルに何れかの LUT Block 内の LUT に配置される必要がある．この制約を導入した変数を用いて表すと次のような式になる．

$$\forall op, \sum_{t}\sum_{p} X(op,t,p)=1$$

1 つの LUT Block に含まれる LUT の個数を n 個とすると，これはある LUT Block では全てのサイクルを通して，回路の全ての入力と出力を除いたノードを n 個までしか配置することはできないことと等しいため，上記のノードをop_nとおくと制約式は次のようになる．

$$\forall p, \sum_{op_n}\sum_{t} X(op_n,t,p)\le n$$

LUT Block に含まれる LUT を n 個としたとき，あるサイクルにおいて LUT Block は n 種類までのデータ移動の中継地点としての動作及びノードの配置が可能となる制約は次のような式となる．

$$\forall t, \forall p, \sum_{i}\sum_{d} D(i,t,p,d)\le n$$

次にノードが配置された際のデータ移動に関する制約式を定義する．あるノード op がサイクル t において LUT Block p に配置されるとき，ノード op の入力データがサイクル t に LUT Block p へ向かう配線上に存在する必要がある．ノード op の入力データをi_{REQ}，LUT Block p へ向かう配線の集合をp_{IN}と置くと，制約式は次のようになる．

$$\forall op,t,p,i_{REQ}, -X(op,t,p)+\sum_{p_d\in p_{IN}} D(i_{REQ},t,p,d)\ge 0$$

同じような制約として，あるノード op がサイクル t において LUT Block p に配置されるとき，ノード op の出力データが次サイクル t+1 に LUT Block p から出ていく配線上に存在する必要がある．この配線には自身に回帰する配線も含まれる．ノード op の出力データをi_{GEN}，LUT Block p から出ていく配線の集合をp_{OUT}と置くと，制約式は次のようになる．

$$\forall op,t,p,i_{GEN}, -X(op,t,p)+\sum_{p_d\in p_{OUT}} D(i_{GEN},t+1,p,d)\ge 0$$

次にあるデータ移動が存在した際の前サイクルにおける挙動に関する制約式を定義する．あるデータ i がサイクル t に配線$p(d)$に存在するとき，前サイクル t-1 では LUT Block p へ向かう配線$p(d)'\in p_{IN}$にデータ i が存在するか若しくは LUT Block p にデータ i を出力に持つノードop_{GEN}が配置されている必要がある．この制約がもつ意味はデータが計算せずに生じることはないことに等しい．LUT Block p へ向かう配線の集合をp_{IN}，LUT Block p から出ていく配線の集合をp_{OUT}と置くと，制約式は次のようになる．

$$\forall i,t,p,5\times\sum_{p_{d'}\in p_{IN}} D(i,t-1,p',d')+5\times X(op_{GEN},t-1,p) - \sum_{p_d\in p_{OUT}} D(i,t,p,d)\ge 0$$

ここで制約式に 5 という係数が記されてあるが，これは方向 d の種類を指している．

同様に，あるデータ移動が存在した際の次サイクルにおける挙動に関する制約式を定義する．あるデータ i がサイクル t に配線$p(d)$に存在するとき，次サイクル t+1 では LUT Block p から出ていく配線$p(d)'\in p_{OUT}$にデータ i が存在するか若しくは LUT Block p にデータ i を入力に持つノードop_{REQ}が配置されている必要がある．この制約がもつ意味は通信されたデータが勝手に消失しないことに等しい．LUT Block p へ向かう配線の集合をp_{IN}，LUT Block p から出ていく配線の集合をp_{OUT}と置くと，制約式は次のようになる．

$$\forall i,t,p, -\sum_{p_d\in p_{IN}} D(i,t,p,d)+X\left(op_{REQ},t,p\right) + \sum_{p_{d'}\in p_{OUT}} D(i,t+1,p',d')\ge 0$$

次に回路の入力を表すノードop_{load}と，出力を表すノードop_{store}に関する制約式を定義する．単位 FPGA では回路の入出力データを保持するメモリ領域が単位 FPGA を取り囲むように存在しているため，op_{load}とop_{store}は必然的にメモリ領域に接した LUT Block に配置される必要がある．メモリ領域に接した LUT Block の集合をp_{MEM}と置くと，制約式は次のようになる．n の値は，単位 FPGA の角の LUT Block p においては 2 となり，それ以外の$p\in p_{MEM}$においては 1 となる．

$$\forall p\in p_{MEM}, \sum_{op_{load}}\sum_{t} X(op_{load},t,p)\le n$$

$$\forall p \notin p_{MEM}, \sum_{op_{load}} \sum_{t} X(op_{load}, t, p) = 0$$

$$\forall p \in p_{MEM}, \sum_{op_{store}} \sum_{t} X(op_{store}, t, p) \leq n$$

$$\forall p \notin p_{MEM}, \sum_{op_{store}} \sum_{t} X(op_{store}, t, p) = 0$$

次に順序回路中の FF の出力を表すノードop_{ffload}と，入力を表すノード$op_{ffstore}$に関する制約式を定義する．内部に FF を持つ回路を連続的に単位 FPGA で動作させるには，ある FF のノード$op_{ffstore}$からノードop_{ffload}へとデータが移動する必要がある．そのため FF のノード$op_{ffstore}$とノードop_{ffload}を同じ LUT Block p に配置し，p の配線$p(L)$を用いてデータを保持し続けるよう設計する．回路を配線可能な最大サイクルを T，LUT Block 間の配線数を E と置くと，制約式は次のようになる．

$$\forall t, \forall p, \sum_{1}^{t-1} \sum_{op_{ffstore}} X(op_{ffstore}, t, p) + \sum_{i} D(i, t, p, L) \leq E$$

$$\forall t, \forall p, \sum_{t}^{T} \sum_{op_{ffload}} X(op_{ffload}, t, p) + \sum_{1}^{t} D(i, t, p, L) \leq E$$

1 つの FF に対し 1 つの LUT Block p の配線$p(L)$を用いるため，LUT Block 内の LUT が n 個だとすると 1 つの LUT Block あたりの配置可能な FF に関するノード数は n 以下であるという制約は次のような式となる．

$$\forall p, \sum_{op_{ffload}} \sum_{t} X(op_{ffload}, t, p) \leq n$$

最後に，LUT Block 間の配線数 E を定義する制約式は次のようになる．

$$\forall t, \forall d, \sum_{i} D(i, t, p, d) \leq E$$

これらの制約によって最適化された結果を求めるために，DFG のすべての終端ノードが配置されるサイクルを求めて最も大きい値T_{MAX}を実行時間とする．つまりこの ILP モデルの目的関数はT_{MAX}であり，この値の最小値を求めることが目標となる．

$$\forall t, p, \sum_{t} \sum_{p} t * X(op, t, p) \leq T_{MAX}$$

$$Minimize: T_{MAX}$$

以上の基本的な制約式に加えて，ここでは省略するが回路の入力を表すノードが一部の LUT Block に集中しすぎることにより配線が混雑し配置が困難になることを防ぐための制約がある．

これらの制約を回路に適用し 4.1 節で後述するコンパイラにより制約ファイルを生成した際のファイルのサイズと生成時間のいくつかを表 1 に示す．比較的小さな回路でも制約式の数は大きくなっている．

回路	回路のLUTの最大入力数	回路のLUT数	FPGAサイズ	生成時間	制約数	ファイルサイズ
s298	5	32	6×6	10s	934,446	136MB
s526	5	63	6×6	18s	1,785,994	259MB

表 1　制約ファイルの生成時間とその大きさ

4. 配置配線手法の実装

4.1 制約式を生成するまでの流れ

この節では ILP ソルバーに渡す ILP 問題が記述されるファイルを生成するまでの流れについて説明する．まずハードウェア設計言語などで記述された回路のソースコードを用意する．そして論理合成ツールである ABC[4]を用いて n 入力 LUT のみで設計できる回路へと変換し，回路の入出力と LUT との関係がデータフローグラフ（DFG）で表された DOT 言語で記述された dot ファイルを出力する．この工程をベンチマーク回路である s27（ISCAS89[5]）を用いて図示したものが図 4 である．

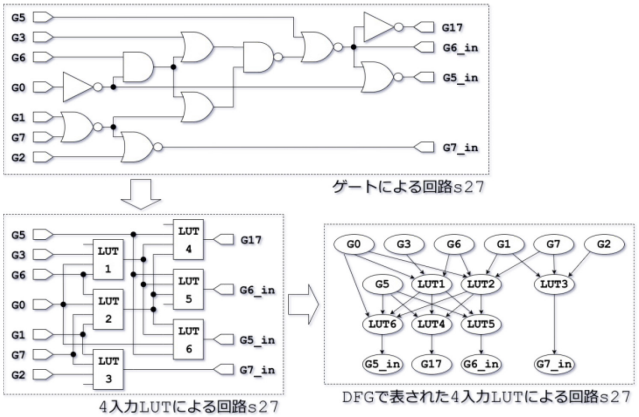

図 4　ABC を用いた回路 s27 の変換フロー

出力された dot ファイルが示す関係に基づいて，DFG のノードとエッジの提案したアーキテクチャ上における配置配線を可能とする制約が記された lp ファイルを生成する．その後得られた lp ファイルを ILP ソルバーに読み込ませることで LP 問題の解を得る．

ISCAS89 ベンチマーク回路 s27 を，5 入力以下の LUT5 個からなる回路に変換した回路を，LUT Block が 1 辺に 6 個存在する 6×6 の単位 FPGA 上に配置配線を試みた結果，約 149 秒で最適解が得られた．解の探索には 5.1 節で後述する計算機上で ILP ソルバーである Gurobi Optimizer[6]を用いた．一方で s298 を 5 入力以下の LUT32 個からなる回路に変換した回路の配置配線は，60,000 秒を超えても許容解すら得ることはできなかった．LUT の数が増えると許容解を得るのですら膨大な時間がかかってしまうことから，まずは LUT の数を増やした際に許容解を得ることができるようにする発見的手法を次の節で提案する．

4.2 より大規模な回路の配置配線を可能とする手法の提案と実装

回路を DFG に置き換えたとき，入出力ではない DFG 上のあるノードについて考える．このノードから入力側に辿っていったとき，最も遠い，つまり多くのエッジを経る必要のある入力ノードとの間のエッジ数をそのノードのレベルと定義する．そして最もレベルの大きいノードのレベル数をその回路のレベルと定義する．例えば図 4 の LUT3 と書かれたノードのレベルは 1, LUT5 と書かれたノードのレベルは 2 となり，回路 s27 のレベルは 2 となる．このように回路をノードのレベルにより分割し，レベル 1 までのノードのみを含むレベル 1 の部分回路から配置配線を行い，得られたノードの配置場所のデータをレベル 2 の部分回路に既知の解として与え配置配線を行う．これらを繰り返すことでより短時間で許容解を得られるのではないかと考えた．このとき全てのレベルについてこの工程を繰り返すと解が得られる時間は短くなるが解の質が下がってしまうことが考えられるので，本研究では回路のレベルを 2 で切り上げ除算したレベルの部分回路まで配置配線することとした．例として図 5 のベンチマーク回路 s208.1（ISCAS89）に対しては回路レベルが 3 であるのでレベル 2 までの部分で回路を分割し配置配線を行う．

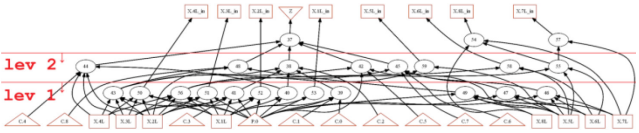

図 5　回路 s208.1 に対する回路分割

この配置配線手法の問題点として，低いレベルの回路を配置配線した際に一部の論理素子資源や配線素子資源が飽和してしまい，より高いレベルの回路を配置配線できない事態が生じることが挙げられる．この問題が生じた際に，より高レベルの回路に既知として与えていた配置場所のデータのサイクルを拡張することでこの問題を解消できると考えた. サイクルを拡張するというのは, 例えば回路 s208.1 のレベル 1 の部分回路が 3 サイクルで配置配線できるという解が得られたとき，レベル 2 の部分回路に与える既知の解の 3 サイクル目を，LUT を配置する LUT Block は変更せずに 4 サイクル目に置き換えることを指す.

4.1 節の DFG で与えられた論理回路のデータを入力とし，制約ファイルを生成したのち ILP ソルバーに与え，今節の配置配線手法を自動的に行うプログラムを Python で記述し設計した. プログラムの大きさは 1,013 行となった. 図 6 は処理の流れを示している．このプログラムでは ILP ソルバーによる計算時間を 1 つのファイル当たり 10,000 秒に制限しており，10,000 秒以内に最適解が見つからない場合はその時点で得られた許容解が解として得られるようになっている.

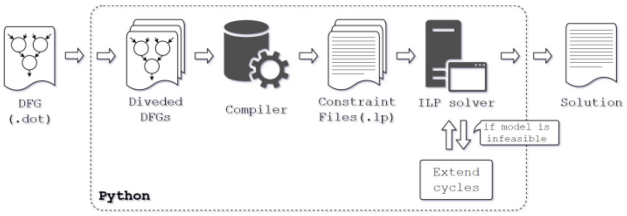

図 6　Python によるプログラムの仕組み

5. ベンチマーク回路を用いた提案手法の評価

5.1 提案手法による配置配線結果と考察

4.2 節で作成したプログラムにより，いくつかの回路を単位 FPGA の仕様を変更しつつ配置配線を行った．使用した回路，単位 FPGA の仕様は以下のとおりである，論理合成ツールとして ABC 1.01 を，ILP ソルバーとして Gurobi Optimizer 7.0.1 を使用した．計算機として CPU が Xeon E5-2699 v4 2.20GHz（1P/22C）×2，RAM が 512GB のものを用いた.

回路：298, s386, s444, s526（ISCAS89）

単位 FPGA の仕様を定義する変数：

LUT の最大入力数：3, 4, 5, 6, 7（既定値 5）

FPGA サイズ: 4×4, 5×5, 6×6, 7×7, 8×8 (既定値 6×6)

LUT Block に含まれる LUT の数 : 3, 4, 5, 6, 7(既定値 5)

LUT Block 間の通信線数 : 1, 2, 3, 4, 5（既定値 5）

単位 FPGA の仕様を定義する変数を 1 つ選択し変化させたときのサイクル数と実行時間を測定した．他の 3 つの変数は上で既定値として記した値で固定した.

表 2 は結果の一部を示している．詳細については参考文献[3]を参照されたい．計算時間は実際に Gurobi Optimizer による解の探索にかかった時間の合計を表しており，制約ファイルの生成などは時間に含まれてはいない.

回路	回路のLUTの最大入力数	回路のLUT数	FPGAサイズ	LUT Block内のLUT数	LUT Block間の通信線数	サイクル数	計算時間
s298	6	24	6×6	5	5	7	423s
	5	32	4×4	5	5	7	17s
	5	32	6×6	5	5	9	249s
	5	32	6×6	5	4	8	156s
s386	7	33	6×6	5	5	7	348s
	5	49	4×4	5	5	8	32s
	5	49	6×6	5	5	9	191s
	5	49	6×6	5	4	8	158s
s444	7	32	6×6	5	5	10	239s
	5	46	4×4	5	5	8	67s
	5	46	6×6	4	5	12	332s
	5	46	6×6	5	2	11	348s
s526	6	50	6×6	5	5	9	326s
	5	63	5×5	5	5	10	459s
	5	63	6×6	5	5	10	727s
	5	63	8×8	5	5	10	2,966s

表 2　実験結果の一部

回路の LUT の最大入力数を変化させた結果, どの実験においても LUT 最大入力数 3 の回路が最もサイクル数が大きくなった一方で，他の入力数の場合に有意差は見られなか

った．入力数 3 のとき最もサイクル数が多くなったのは，回路の段数が他の入力数のものよりも多かったこと，加えてノード数が増えたことにより配置場所の混雑が生じたからであると考えられる．

単位 FPGA のサイズを変化させたときの結果については，今回の実験で使用した回路中の LUT 数が比較的少なかったこともあり，単位 FPGA 上への配置配線は 1 辺あたりの LUT Block 数が解が得られたもののうち最も小さいサイズのときにより少ないサイクル数となった．この結果から，単位 FPGA では LUT 使用率が 0.4 を超えるようなサイズが最適ではないかと推測できる．

LUT Block 内の LUT 数と LUT Block 間の通信線数を変化させたときの結果は，それぞれ LUT 数が多くなるほど，また通信線数が多くなるほどサイクル数は少なくなると予想していたが必ずしもそうはならなかった．これは LUT 数が増えたり通信線数が増えたりすることで低レベルの回路における配置の自由度が増えてしまい混雑した配置になってしまった結果，高レベルの回路の配置時における配線の自由度が減ってしまったからではないかと考えられる．

5.2 より大規模な回路の配置配線結果

ISCAS89 に含まれる，本実験で用いた回路よりも大きな回路の配置配線を試みた際の結果は以下の表 3 のようになる．この結果より現時点では最大入力数が 5 以下の LUT が 109 個以上の回路は配置配線不可能となっている．配置配線が可能であった回路でもサイクル数が前節の実験で得られたサイクル数より大幅に悪化しているものもあり，実用的なサイクル数で配置配線可能な回路は最大入力数が 5 以下の LUT が 63 個以下のものに限られると推測される．

回路	回路のLUTの最大入力数	回路のLUT数	FPGAサイズ	LUT Block内のLUT数	LUT Block間の通信線数	サイクル数	計算時間
s820	5	99	6×6	5	5	21	7,908s
s832	5	101	6×6	5	5	15	32,747s
s838.1	5	108	6×6	5	5	20	19,887s
s953	5	145	7×7	5	5	x	x

表 3　より大規模な回路の実験結果

6. おわりに

本研究では，従来の FPGA におけるタイミングクロージャ問題を解決するための FPGA をモデル化し，モデル上への回路の配置配線を整数線形計画問題として扱うことで可能とする手法を提案した．また提案手法を Python により実装することで，回路規模が比較的小さい 60 個程度の 5 入力 LUT からなる回路までは配置配線を短時間で行うことに成功した．そして，回路を配置配線する FPAG をその仕様を変えて実験を行い，隣接するブロック間のみに配線を持つ FPGA として最適なアーキテクチャについて調べようとした．単位 FPGA のサイズについては，LUT Block 内の LUT 数を 5 個，通信線数を 5 本とした場合では FPGA 上で配置可能な LUT 数における回路中の LUT 数の割合が 0.4 を超えるサイズのときに最小サイクル数で配置配線可能となった．他のアーキテクチャに関するパラメータについては，結果が様々であり有意な結果を得ることができなかった．

本研究におけるサイクル数の評価は，実際に単位 FPGA を製作していないため定量的なものとはならない．しかし実際の FPGA に回路を配置配線した際のクリティカルパスの遅延時間が，本実験で得られたサイクル数に 2.2 節の最大配線長の遅延時間を乗算したものより大きければ，本研究で想定した単位 FPGA 上の方がより高速に回路をハードウェアとして実装できるといえる．

本研究で想定した単位 FPGA は，LUT Block 内に複数の LUT を含むため従来の FPGA よりも同じブロック数であったとしても FPGA のサイズは大きくなると考えられる．そのため LUT Block 内に例えば 5 入力 LUT を 5 個配置する代わりに[7]で示されているような 5 入力 5 出力のメモリを配置することにより LUT Block の面積を減少させる方法が考えられる．このようなメモリを配置するということは，5 個の 5 入力 LUT の入力がすべて同じ組み合わせになることに等しいため，本研究で想定した単位 FPGA とは異なってしまう．しかしこの条件を整数線形計画問題として立式することは可能であり，本実験で用いたコンパイラを拡張することで容易に実装することができる．

現時点での課題として，本実験に用いた回路に比べより大規模な回路が短時間のうちに実装不可能となっていることがまず挙げられる．このためより大規模な回路を，サイクル数を増やさずに配置配線を可能にする手法を考案する必要があると考えている．また，より少ないサイクル数での回路の配置配線を実現させるための手法としてパイプライン実行が考えられる．パイプライン化により，レイテンシは変化しないか少量の増加でスループットを小さくできると考えられるため，今後検討していきたい．また，積和計算のように最適なスケジューリングがわかっている問題に適用して，得られる解の最適性についての解析も行っていきたい．

参考文献

[1] Yi Lu, "Communication Aware Compiler for Mesh-Structured Reconfigurable Processors". 東京大学大学院工学系研究科修士論文, 2016.
[2] Yi Lu, Qinhao Wang, Amir Masoud Gharehbaghi, Masahiro Fujita, Communication Aware Compiler for Mesh-Structured Reconfigurable Processors on Single/Multi Chip, 31st International Technical Conference on Circuits/Systems, Computers and Communications, Okinawa, July 2016.
[3] 丸岡 大浩, “隣接するブロック間のみに配線をもつ FPGA に対する配置配線手法”. 東京大学工学部卒業論文, 2017.
[4] ABC: A System for Sequential Synthesis and Verification, https://people.eecs.berkeley.edu/~alanmi/abc/.
[5] ISCAS89, http://www.pld.ttu.ee/~maksim/benchmarks/iscas89/.
[6] Gurobi Optimization, http://www.gurobi.com/index.
[7] 太陽誘電株式会社, 再構成可能な論理デバイス. 特公 2014-163099, 2014-04-02.