Vol.2016-ARC-219 No.31
Vol.2016-SLDM-175 No.31
Vol.2016-EMB-40 No.31
2016/3/25

RQFPゲートを用いた超低消費電力AQFP論理回路の設計手法

松本 涼平[1,a)]　山下 茂[1]　竹内 尚輝[2]

概要：近年，超電導集積回路技術が注目されており，そのなかでも特に低消費電力性という観点において，AQFP (Adiabatic Quantum-Flux-Parametronron) 論理回路が注目されている．本稿では，AQFP 論理回路分野のなかでも，可逆的に動作する 3 入力 3 出力の RQFP (Reversible Quantum-Flux-Parametronron) ゲートに注目した．RQFP ゲートは，3 出力のすべてを次段のゲートに接続しないと可逆性を失い，ゲートの消費電力が増加する．また，RQFP ゲートの 3 出力の出力論理は，それぞれ対応する 1 入力を論理否定した多数決論理という特殊な論理である．RQFP ゲートの特殊な出力論理を考慮し，また RQFP ゲートのみを用いて AQFP 論理回路を設計する手法はまだ存在しない．そこで，多数決論理の入力の値を特定の値で固定することで基本論理が表現できることに注目し，RQFP ゲートのみを用いてあらゆる論理関数を表現する論理回路を設計する手法として，出力論理の一致を利用して回路内の未接続出力を削減する手法を提案した．また，CSPF (Compatible Set of Permissible Functions) という概念を利用して回路内の未接続出力を削減する手法も提案した．CSPF を用いた提案手法では，入力回路を単純に RQFP ゲートのみの AQFP 論理回路に変換するだけの手法や，出力論理の一致を利用する提案手法に比べて，それぞれ平均で約 47.71%，20.95%の未接続出力の数が削減できた．

1. はじめに

将来のエクサスケール級を超える高性能コンピュータの実現のためには，エネルギー効率の高い論理演算回路の実現が必要となると考えられる [1]．論理演算回路のエネルギー効率において，最も重要な性能指標は 1 ビットの計算あたりの消費エネルギーである．そのため，古くからこの最小エネルギーに関する研究が行われてきた．理論的検討によれば，1 ビットの計算に必要な最小エネルギーは雑熱音エネルギー程度と予想され，Landauer 限界と呼ばれている [2]．現在広く使用されている CMOS 回路を用いたコンピュータでは，この最小エネルギーよりはるかに大きなエネルギーで計算が行われている．そのため，消費エネルギーの限界を議論する必要がなかった．しかし，リーク電流の増大による消費電力の増加や放熱といった問題から CMOS 回路における性能向上の限界が叫ばれつつあるため，消費エネルギーの限界についても議論されるようになってきた．

そこで近年，高速で動作し，低消費電力性に優れた超伝導集積回路技術が注目されている．そのなかでも，特に低消費電力性という観点において，可逆的量子磁束パラメトロン (Reversible Quantum-Flux-Parametronron: RQFP) ゲート [5] が注目されている．RQFP ゲートは，超伝導回路の一種である断熱型量子磁束パラメトロン (Adiabatic Quantum-Flux-Parametronron: AQFP) [4] を可逆的に動かすことにより実現される 3 入力 3 出力のゲートである．可逆的に動作することにより，RQFP ゲートが演算で消費するエネルギーは，Landauer 限界よりも少ない低エネルギーとなる．ただし，RQFP ゲートを用いて複雑な AQFP 論理回路を自動設計する手法はまだ存在しない．

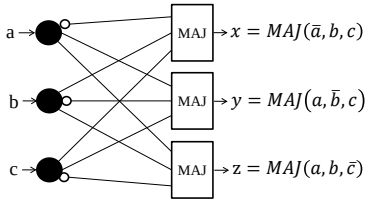
RQFP ゲートは，可逆的に動作させることにより情報のエントロピーを消失せず，超低消費電力で動作する．しかし，回路内で RQFP ゲートと同時に非可逆なゲートを用いたり，RQFP ゲートの一部の出力をどこにも接続せずその RQFP ゲートの可逆性を損なわせたりすると，消失する情報のエントロピー分の消費電量が増大してしまう．

本稿では，RQFP ゲートのみを用い，かつ可能な限り RQFP ゲートの接続しない出力を削減した AQFP 論理回路を自動設計する手法を提案する．提案手法では，AND/NOT ゲートのみで構成された論理回路のすべてのゲートを RQFP ゲートに変換することにより RQFP ゲートのみの AQFP 論理回路を設計する．また, 各 RQFP ゲートの接続しない出力は，CMOS 回路の面積を削減する手法であるトランスダクション法 [3] にて紹介されている，ドントケアを考慮した出力関数をあらわす CSPF (Compatible Set of Permissible Functions) を用いて可能な限り削減する．

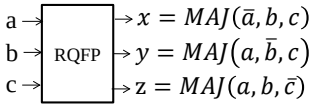
生成回路内の RQFP ゲートの未接続出力について，削減しない手法（単純手法），出力論理の一致を利用して削減する手法（提案手法（論理一致）），CSPF を用いて削減する手法（提案手法（CSPF））の 3 つの手法をいくつかのベンチマーク回路の設計に適用した．その結果，提案手法（CSPF）で生成された回路内の未接続出力の数は，単純手法および提案手法（論理一致）に比べて，それぞれ平均で 47.71%, 20.95%削減された．

[1] 立命館大学
[2] 横浜国立大学
[a)] marimo@ngc.is.ritsumei.ac.jp

Vol.2016-ARC-219 No.31
Vol.2016-SLDM-175 No.31
Vol.2016-EMB-40 No.31
2016/3/25



(a) RQFP ゲートの構成



(b) RQFP ゲートの表記

図 1: RQFP ゲート

2. 準備

2.1 AQFP 論理回路と RQFP ゲート

2.1.1 動作原理と基本素子

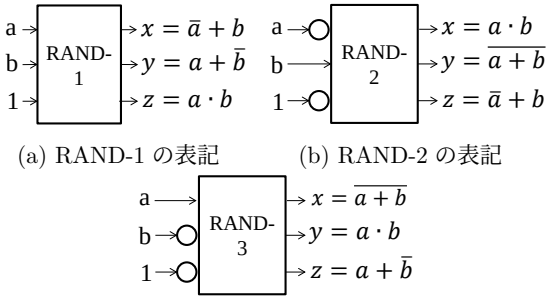
AQFP 論理回路で使用される素子において，出力の論理値は，入力される電流に対して励磁電流が流れることにより確定する．この励磁電流の役割は，クロック入力と同じであると考えることができる．そのため，AQFP 論理回路における論理値の表現方法は，クロック同期式であると考えることができる．したがって，AQFP 論理回路では，バッファ素子による信号の到着タイミング調整が必要となる．

本稿の AQFP 論理回路では，BUF (Buffer), SPL (SPLitter), RQFP ゲートの 3 つの素子を使用する．BUF は，AQFP 論理回路の基本となる素子である．信号の増幅や信号の到達タイミングの調整に用いられる．また，SPL や RQFP ゲートの構築にも使用される．その際，BUF において，入力インダクタンスと出力インダクタンスとを結合させる向きを反転するだけで論理状態を否定できる．したがって，AQFP 論理回路における論理状態の否定は，後述の RQFP ゲートの入出力部で容易に実現できる．SPL は，入力端子から入力されたデータを分岐し，出力端子に伝搬する分岐素子である．3 出力の SPL (3-SPL) は RQFP ゲートの構築にも使用される．

RQFP ゲートは，3 つの 3-SPL と 3 つの多数決 (Majority: MAJ) ゲートより構成される，3 入力 3 出力の可逆論理ゲートである（図 1）．図 1 (a) において，白丸は論理状態が否定されていることをあらわす．また，MAJ ゲートの出力論理関数は，$MAJ(a,b,c) = ab + bc + ca$ である．RQFP ゲートの出力論理関数は，式 (1) のとおりである．

$$\begin{aligned} F(a,b,c) &= (MAJ(\bar{a},b,c), MAJ(a,\bar{b},c), MAJ(a,b,\bar{c})) \\ &= (x,y,z) \end{aligned} \tag{1}$$

AQFP 論理回路において，使用されている RQFP ゲートはそれぞれ 3 つの出力を持つが，接続先をもたない出力が存在する場合もある．以降，どの素子にも接続されていない出力は，未接続出力と呼ぶことにする．RQFP ゲートでは，未接続出力が存在すると，その分の情報のエントロピーが消失する．つまり，その RQFP ゲートは可逆性を失うため，存在する未接続出力の数に比例して回路の消費電力が増加する．



(a) RAND-1 の表記　(b) RAND-2 の表記

(c) RAND-3 の表記

図 2: RAND

2.1.2 RQFP ゲートによる基本論理の表現

前述したとおり，NOT 論理は RQFP ゲートの入出力に自由に組み込むことができる．また，RQFP ゲートは，ある 1 つの入力の論理値を 0 または 1 に固定することにより，NOT 論理以外の基本論理もあつかうことができる．以降，RQFP ゲートを用いた AND 論理を表現するゲートを RAND と呼ぶことにする．

RAND は，RQFP ゲートの 1 入力の論理値を 1 に固定するか，1 入力の論理値を 0 に固定してなおかつ対応する 1 入力を論理否定することにより実現できる．たとえば，入力 c の論理値を 1 に固定することで出力 z の論理が AND 論理となる（図 2 (a)）．以降，この方法で作成するゲートを RAND-1 と呼ぶ．また，同様に入力 c の論理値を 0 に固定し，入力 a あるいは b の論理値を論理否定することで出力 x あるいは y の論理が AND 論理となる（図 2 (b), (c)）．以降，この方法で作成するゲートをそれぞれ RAND-2，RAND-3 と呼ぶ．また，RAND において，AND 論理を出力する出力（RAND-1, 2, 3 のそれぞれ z, x, y 出力）を AND 出力部と呼ぶ．

2.2 トランスダクション法

ゲートまたは結線の論理関数を論理関数 f で置き換えても，回路のすべての出力の定義されている部分に変化がないような論理関数 f を，ゲートまたは結線の許容関数 [3] と呼ぶ．許容関数は 0, 1, $*$（ドントケア）の 3 値で表される．特に，同時に置き換え可能な許容関数の集合を CSPF (Compatible Set of Permissible Functions) と呼び，ゲート i の CSPF は $CSPF(i)$ とあらわす．CSPF の詳しい計算方法については [3] に詳しく記載されている．

トランスダクション法では，CSPF を用いて回路の結線のつなぎ換えを行える．ここでは，本稿で用いる CSPF による回路の結線のつなぎ換えの手法について述べる．以下の条件 1, 2 が成り立てば，ゲート g_i の出力を外部入力またはゲート g_j の出力で置き換え可能である．

1. $f(g_j) \in CSPF(g_i)$ を満たす．
2. g_j から g_i へのパスが存在しない．

ただし，この手法でゲートを置き換えた場合，接続先をもたないゲートが出現する可能性がある．そのゲートは外部出力に影響を及ぼさないため，削除することができる．

3. RQFP ゲートを用いた AQFP 論理回路の設計法

3.1 問題定義

前述したように，RQFP ゲートを用いた AQFP 論理回

Vol.2016-ARC-219 No.31
Vol.2016-SLDM-175 No.31
Vol.2016-EMB-40 No.31
2016/3/25

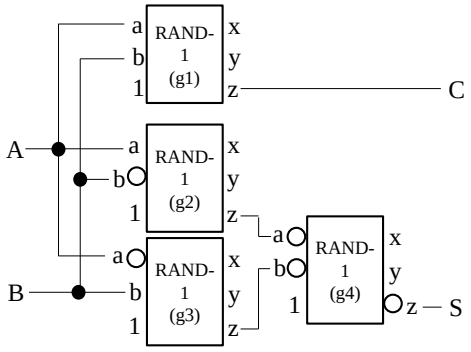


図 3: RAND-1 のみの AQFP 論理回路

路では，論理演算のための回路として RQFP ゲートしか用いないこと，およびその RQFP ゲートの未接続出力をできるだけ少なくすることが消費電力の削減に関して重要となる．その点を考慮し，本稿では，以下のように問題を定義する．

- 入力：AND/NOT ゲートのみで構成される組み合わせ回路
- タスク：以下の制約を満たしながら，入力回路を AQFP 論理回路に変換する．
  - RQFP ゲート以外の論理ゲートが論理回路内に存在してはいけない．
  - RQFP ゲートの未接続出力はできるだけ削減する．
  - AQFP 論理回路はパイプライン構造をとるため，段数が後ろのゲートの出力から手前のゲートの入力へ接続してはいけない．

3.2 出力論理の一致を利用した未接続出力の削減手法

提案手法では，消費電力をできる限り少なくするために，使用する RQFP ゲートの未接続出力をできるだけ削減する．ただし，計算量の爆発を防ぐために，RQFP ゲートの未接続出力を削減する過程で新たな RQFP ゲートを追加することは禁止する．

3.2.1 出力論理の一致を利用した提案手法の概要

提案アルゴリズムの全体の流れを以下に示す．

ステップ 1　入力回路を RAND-1 のみの AQFP 回路に変換
ステップ 2　合成された回路の未接続出力を削減

ステップ 1 では，入力される AND/NOT ゲートのみで構成される回路を，前述した図 2 (b) の RAND-1 を用いて AQFP 論理回路に変換する．その際，入力回路に存在する NOT ゲートは，RAND-1 で吸収する．図 3 は，RAND-1 を用いて AND/NOT のみで構成された半加算器を AQFP 論理回路に変換した例である．図 3 において，黒丸は SPL，白丸は論理否定をあらわす．AND/NOT のみで構成された半加算器に存在する NOT ゲートは，図 3 の回路において，それぞれ $g2$ の入力 b, $g3$ の入力 a, $g4$ の入力 a と b および出力 z にそれぞれ吸収されている．また，図 3 の回路において，4 つの RQFP ゲートの出力 x, y がそれぞれ未接続であるため，未接続出力は合計 8 つ存在する．

ステップ 2 は，ステップ 1 で生成される回路の未接続出力を削減する提案手法の主要な部分となるステップである．このステップでは，ゲートが別のゲートの出力で置き換え可能かどうか，入力回路内を外部出力側から外部入力側に向かって探索し，可能であればその置き換えを行う．その際，RQFP ゲートを新たに回路内に追加するような置き換えは行わない．

Algorithm 1 OptimizeCircuit：未接続出力を削減（ステップ 2）

Require: $UOFuncTable$：C 内のすべてのゲートにおける未接続出力の情報
Ensure: C 内のそれぞれのゲートの出力論理は計算済み
1: $UOFuncTable$ を生成
2: **for all** 外部出力リストのゲート (po) **do**
3:　**for all** po の入力ゲート (po_in) **do**
4:　**if** po_in を未探索 **then**
5:　OptimizeSubCircuit(po_in)　▷ Algorithm 2
6:　**end if**
7:　**end for**
8: **end for**

3.2.2 未接続出力の削減アルゴリズム

ステップ 2 で用いられる本アルゴリズムでは，未接続出力を削減するため，まずゲートの出力論理が別のゲートにおける出力の出力論理と等しいかどうかを調べる．そして，もしそのような出力論理が一致する組み合わせが存在すれば，別のゲートの出力を利用してゲートを置き換える．

Algorithm 1 および Algorithm 2 に，未接続出力の削減アルゴリズムを示す． 本アルゴリズムでは，はじめに，未接続出力を削減するために必要な論理関数の情報をテーブル（以降，$UOFuncTable$）にメモしておく．ここでメモした論理関数は，RQFP ゲートを置き換えられるかどうかを CanReplaceGate により判定する際に使用する．

次に，回路 C 内を外部出力側から外部入力側に向かって再帰的に探索する．その際に呼び出される Algorithm 2 では，まず引数となる g の出力論理と置き換え可能な別の RQFP ゲートの出力が存在するかどうか判定し，置き換え可能であれば置き換えを行う（15-17, 22-24, 31-33 行目）．

もし RQFP ゲートの置き換えを行った場合，$UOFuncTable$ の更新を行う（38-40 行目）．最後に，g の入力ゲートを引数として，Algorithm 2 を外部入力ゲートに到達するまで再帰的に呼び出す．

3.2.2.1 必要な未接続出力の情報

Algorithm 1 の 1 行目では，ゲート i について以下の項目の情報を $UOFuncTable$ にメモする．

1. ゲート i における AND 出力部の出力論理
2. ゲート i (RAND-1, 2, 3) における AND 出力部以外の 2 つの出力論理
3. 上記の 7 つの出力論理を否定した出力論理

3.2.2.2 CanReplaceGate

CanReplaceGate は，ゲートの出力が別のゲートの出力で置き換えられるかどうかを判定するメソッドである．CanReplaceGate のアルゴリズムを Algorithm 3 に示す．$idepth$ は，ゲートの外部入力からの段数である．たとえば，図 3 において，$g1$ と $g4$ の $idepth$ はそれぞれ 1 と 2 である．$target\_gate$ は，g の出力と同じ出力論理を持つことから g と置き換えられる可能性を持つゲートの出力のリストである．たとえば，ゲート i の出力 y などを意味する．

Algorithm 3 では，まず g の $idepth$ が，g と置き換えられる可能性を持つ出力を持つゲートの $idepth$ よりも小さいかを調べる（5-8 行目）．g が置き換えを考えるゲートよりも前段にある場合，置き換えはできないため $false$ が返される．また，g を自身の出力で置き換えることもできないため，その場合も $false$ が返される（9-11 行目）．g の接続先を持つ出力が 1 つの場合，この時点で置き換え可能であるとわかるため，$true$ が返される（14 行目）．

Algorithm 2 OptimizeSubCircuit

Require: g：Algorithm 2 への入力ゲート
Require: g_out_i：g の出力部 $(1 \leq i \leq 3)$
Require: $check\_f_i$：$UOFuncTable$ 内の g_out の出力論理と一致する情報を格納するリスト型の配列 $(1 \leq i)$
1: **if** g が外部入力ゲート **then**
2: **return**
3: **end if**
4: $i \leftarrow 1$
5: **for all** 接続先が存在する g_out (tmp_g_out) **do**
6: **for all** $UOFuncTable$ の情報 (UO_f) **do**
7: **if** tmp_g_out の出力論理 $= UO\_f$ **then**
8: $check\_f_i$ の最後に UO_f をプッシュ
9: **end if**
10: **end for**
11: $i \leftarrow i + 1$
12: **end for**
13: **if** $check\_f$ のサイズ $= 1$ **then**
14: **for all** $check\_f_1$ (UO_f_1) **do**
15: **if** CanReplaceGate(g, $check\_f$ のサイズ, UO_f_1) $=$ **true then**
16: g を置き換え可能な別のゲートの出力で置き換え，38 行目にジャンプ
17: **end if**
18: **end for**
19: **else if** $check\_f$ のサイズ $= 2$ **then**
20: **for all** $check\_f_1$ (UO_f_1) **do**
21: **for all** $check\_f_2$ (UO_f_2) **do**
22: **if** CanReplaceGate(g, $check\_f$ のサイズ, UO_f_1, UO_f_2) $=$ **true then**
23: g を置き換え可能な別のゲートの出力で置き換え，38 行目にジャンプ
24: **end if**
25: **end for**
26: **end for**
27: **else if** $check\_f$ のサイズ $= 3$ **then**
28: **for all** $check\_f_1$ (UO_f_1) **do**
29: **for all** $check\_f_2$ (UO_f_2) **do**
30: **for all** $check\_f_3$ (UO_f_3) **do**
31: **if** CanReplaceGate(g, $check\_f$ のサイズ, UO_f_1, UO_f_2, UO_f_3) $=$ **true then**
32: g を置き換え可能な別のゲートの出力で置き換え，38 行目にジャンプ
33: **end if**
34: **end for**
35: **end for**
36: **end for**
37: **end if**
38: **if** g を別のゲートに置換した場合 **then**
39: UpdateUOFuncTable()
40: **end if**
41: **for all** g の入力ゲートリスト (g_in) **do**
42: OptimizeSubCircuit(g_in) ▷ Algorithm 2
43: **end for**

次に，g が接続先を持つ出力部を複数持つ場合の置き換え条件について考える（16-26 行目）．g と置き換えられる出力を持つゲートは，置き換え後に RAND-1，RAND-2，RAND-3 のいずれかで固定される．そのため，g の出力論理と一致する出力論理を持つ出力が 2 つ以上同じゲートからの出力である場合，それらのゲートの RAND のタイプは同じでなければならない．これらの条件が満たされる場合に $true$ を返す．

3.2.2.3 ゲートの置き換え方法

g の出力先への接続を，CanReplaceGate により判定した置き換え可能な出力により置き換える．その際，必要に応じて置き換え可能な出力を持つゲートのタイプ（RAND-1，RAND-2，RAND-3）を変更する．置き換え後，g は出力をもたなくなるため，g を削除する．ただし，g の入力ゲートが外部入力でない場合，g が削除されることにより，g の入力ゲートも出力をもたなくなる可能性がある．その場合，その入力ゲートも削除する．このように，ゲートの削除はできなくなるまで再帰的に繰り返す．

Algorithm 3 CanReplaceGate：ゲートが別のゲートの出力で置き換え可能かどうか判定

Require: g, $check\_f$ のサイズ, UO_f_i：Algorithm 3 に渡される引数
Require: $idepth$：ゲートの外部入力からの段数
Require: $target\_gate_i$：置換対象となるゲートの出力のリスト
Require: N：置き換えできる可能性を持つ g_out の数
Ensure: C 内のすべてのゲートの $idepth$ は計算済み
Ensure: C 内のそれぞれのゲートの出力論理は計算済み
1: $N \leftarrow check\_f$ のサイズ
2: **for** $i = 1$ to N **do**
3: $target\_gate$ の最後に UO_f_i の出力論理を持つゲートの出力をプッシュ
4: **end for**
5: **for** $i = 1$ to N **do**
6: **if** g の $idepth < target\_gate_i$ を持つゲートの $idepth$ **then**
7: **return false**
8: **end if**
9: **if** $g = target\_gate_i$ を持つゲート **then**
10: **return false**
11: **end if**
12: **end for**
13: **if** $N = 1$ **then**
14: **return true**
15: **end if**
16: **if** $target\_gate$ を持つゲート同士がすべて異なるゲート **then**
17: **return true**
18: **else if** $target\_gate$ を持つゲート同士がすべて同じゲート **then**
19: **if** $target\_gate$ を持つゲートの RAND のタイプがすべて同じ **then**
20: **return true**
21: **end if**
22: **else if** $target\_gate$ のゲートのうち 2 つが同じゲート **then**
23: **if** 同じゲートからの出力同士の RAND のタイプが同じ **then**
24: **return true**
25: **end if**
26: **end if**
27: **return false**

表 1: 図 3 の $UOFuncTable$

ゲート名	AND 出力部	RAND-1 (x)	RAND-1 (y)	RAND-2 (y)	RAND-2 (z)	RAND-3 (x)	RAND-3 (z)
$g1$	$A \cdot B$	$\overline{A}+B$	$A+\overline{B}$	$\overline{A+B}$	$\overline{A}+B$	$\overline{A+B}$	$A+\overline{B}$
(否定)	$\overline{A}+\overline{B}$	$A \cdot \overline{B}$	$\overline{A} \cdot B$	$A+B$	$A \cdot \overline{B}$	$A+B$	$\overline{A} \cdot B$
$g2$	$A \cdot \overline{B}$	$\overline{A}+\overline{B}$	$A+B$	$\overline{A} \cdot B$	$\overline{A}+\overline{B}$	$\overline{A} \cdot B$	$A+B$
(否定)	$\overline{A}+B$	$A \cdot B$	$\overline{A+\overline{B}}$	$A+\overline{B}$	$A \cdot B$	$A+\overline{B}$	$\overline{A+\overline{B}}$
$g3$	$\overline{A} \cdot B$	$A+B$	$\overline{A}+\overline{B}$	$A \cdot \overline{B}$	$A+B$	$A \cdot \overline{B}$	$\overline{A}+\overline{B}$
(否定)	$A+\overline{B}$	$\overline{A+\overline{B}}$	$A \cdot B$	$\overline{A}+B$	$\overline{A+\overline{B}}$	$\overline{A}+B$	$A \cdot B$
$g4$	$(A+\overline{B})(\overline{A}+B)$	$\overline{A}+B$	$A+\overline{B}$	0	$\overline{A}+B$	0	$A+\overline{B}$
(否定)	$\overline{A} \cdot B + A \cdot \overline{B}$	$A \cdot \overline{B}$	$\overline{A} \cdot B$	1	$A \cdot \overline{B}$	1	$\overline{A} \cdot B$

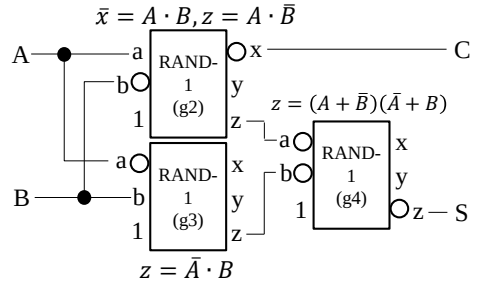
3.2.2.4 UpdateUOFuncTable

UpdateUOFuncTable は，ゲートの置き換えを行った際に必要な $UOFuncTable$ の更新を行うメソッドである．ゲートの置き換えを行うと，置き換え後のゲートの RAND のタイプ (RAND-1, RAND-2, RAND-3) が固定される可能性がある．その場合，固定されるタイプ以外のタイプの出力論理を，$UOFuncTable$ から削除する．

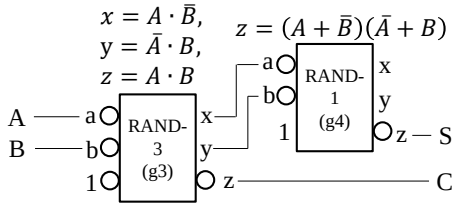
3.2.3 AQFP 論理回路の設計例

本項では, 本節で説明した提案手法について, 図 3 の回路を用いてその適用例についてステップ 2 から説明する．ステップ 2 として, 未接続出力を削減するアルゴリズムをこの回路に適用する．OptimizeCircuit では，まず，$UOFuncTable$ を作成する．表 1 は，作成される $UOFuncTable$ である．次に，回路内で使用されている RQFP ゲートの未接続出

Vol.2016-ARC-219 No.31
Vol.2016-SLDM-175 No.31
Vol.2016-EMB-40 No.31
2016/3/25



(a) $g1$ を $g2$ (RAND-1) の出力 $\bar{x}$ で置き換え



(b) $g2$ を $g3$ (RAND-3) の出力 $\bar{z}$, x で置き換え

図 4: 未接続出力の削除

力を削減するため，回路を外部出力側から外部入力側に向かい探索する．

はじめに $g1$ の置き換えについて考える．$g1$ の出力論理 $A \cdot B$ について $UOFuncTable$ を探索すると，はじめに $g1$ の AND 出力部の出力論理と一致するが，自身のゲートを自身のゲートで置き換えることはできない．再度 $UOFuncTable$ を探索すると，次に $g2$ (RAND-1) の出力 $\bar{x}$ と一致する．$g1$ と $g2$ は異なるゲート同士であり，また，$g1$ と $g2$ の $idepth$ は共に 1 で同じであるため，$g1$ は $g2$ より前段に存在しない．したがって，$g1$ の出力 z は $g2$ (RAND-1) の出力 $\bar{x}$ で置き換えられる．置き換え後の回路を，図 4 (a) に示す．その後，$UOFuncTable$ を更新する．$g1$ は削除されたため，$g1$ に関係する出力論理はすべて削除する．また，$g2$ は RAND-1 に固定されるため，$g2$ (RAND-2, RAND-3) の出力論理もすべて削除する．

次に，$g4$ の置き換えについて考えるが，$g4$ の出力論理 $(A+\overline{B})(\overline{A}+B)$ は $UOFuncTable$ 内に存在しないため，$g4$ は置き換えできない．次に置き換えを考える $g2$ は，出力論理 $A \cdot B$ と $A \cdot \overline{B}$ との 2 つの出力を持つ．$UOFuncTable$ について $g2$ 自身のゲートの出力論理を除いて探索すると，それぞれ $g3$ (RAND-3) の出力 $\bar{z}$ および x と一致する．$g2$ と $g3$ は異なるゲート同士であり，また，$g2$ と $g3$ の $idepth$ は共に 1 で同じであるため，$g2$ は $g3$ より前段に存在しない．さらに，$g2$ の 2 出力の出力論理と一致する $g3$ の 2 出力の出力論理は，どちらも RAND-3 タイプの $g3$ からの出力論理であり，また，それぞれ出力部が z と x で異なる．したがって，$g2$ の出力 x と z は，それぞれ $g3$ (RAND-3) の出力 $\bar{z}$ と x とで置き換えられる．置き換え後の回路を，図 4 (b) に示す．その後，$UOFuncTable$ を更新する．$g2$ は削除されたため，$g2$ に関係する出力論理はすべて削除する．また，$g3$ は RAND-3 に固定されるため，$g3$ (RAND-1, RAND-2) の出力論理もすべて削除する．

$g3$ は明らかに置き換えられないため，図 4 (b) の回路はこれ以上最適化できない．したがって，図 4 (b) の回路が最終的な生成回路となる．最適化前の図 3 の回路では未接続出力は合計で 8 つ存在したが，生成回路では合計で 2 つまで削減できた．

Algorithm 4 Proposed method using CSPF

Require: C：生成回路となる AQFP 論理回路
1: $C \leftarrow$ RAND-1 を用いて AQFP 論理回路に変換された入力回路
2: **repeat**
3: C 内のそれぞれのゲートの出力論理を計算
4: OptimizeCircuitUsingCSPF(C)
5: **until** C の構成に変更なし
6: **return** C

3.3 CSPF を用いた未接続出力の削減手法

3.2 節の提案手法では，ゲート i の出力がゲート j の出力で置き換えられるかどうかについて，出力論理が完全に一致するかどうかで判定していた．しかし，出力論理同士が完全に一致していなくても，ゲート i の CSPF がゲート j の出力論理を包含していれば，ゲート i の出力はゲート j の出力で置き換えられる．

3.3.1 CSPF を用いた提案手法の概要

提案アルゴリズムの全体の流れを Algorithm 4 に示す．CSPF を用いた提案手法の流れは，3.2 節の提案手法の流れと基本的に同じである．3.2 節の提案手法と異なる点は，CSPF の計算が必要な点，ゲートの置き換え判定に CSPF を用いる点である．また，CSPF を用いてゲートの出力の置き換えやゲートの削除を行うと，外部出力の出力論理に影響のない範囲で中間ゲートの出力論理が変更される可能性がある．そのため，一度 CSPF を用いた未接続出力を最適化するアルゴリズムを適用した論理回路でも，再度同じアルゴリズムを適用するとさらに未接続出力を削減できる可能性がある．したがって，Algorithm 4 では，論理回路が変形されなくなる（未接続出力が減らなくなる）まで，未接続出力を削減する最適化アルゴリズム（OptimizeCircuitUsingCSPF）を C に繰り返し適用する．

3.3.2 CSPF を用いた提案手法における未接続出力の削減アルゴリズム

OptimizeCircuitUsingCSPF は，Algorithm 1 の OptimizeCircuit において，はじめに回路内のすべてのゲートの CSPF を計算するステップを追加し，呼び出す OptimizeSubCircuit を OptimizeSubCircuitUsingCSPF に変更したメソッドである．OptimizeSubCircuitUsingCSPF は，Algorithm 2 の OptimizeSubCircuit とほぼ同じである．異なる点は，7 行目における置換可能性を持つ出力論理の判定において，出力論理同士が完全に一致するかどうか調べる代わりに，CSPF が出力論理を包含するかどうかを判定条件として使用する点である．

4. 実験結果・考察

提案手法について評価を行うために，入力回路を RAND-1 のみの回路に変換するだけの手法（単純手法），出力論理の一致を利用した未接続出力の削減手法（提案手法（論理一致））, CSPF を用いた未接続出力の削減手法（提案手法（CSPF））の 3 つの手法について C++言語で実装した．いくつかのベンチマーク回路 [6] の設計において，単純手法，提案手法（論理一致），提案手法（CSPF）をそれぞれ適用した．

実験結果を表 2 に示す．UO 数は生成回路内の未接続出力 (Unused Output) の合計数，時間は計算時間（秒），UO 比 1 (および 2) は提案手法（CSPF）における UO 数の単純手法（および提案手法（論理一致））における UO 数に対する増減率をそれぞれ示す．また，合計の項目の値は，

Vol.2016-ARC-219 No.31
Vol.2016-SLDM-175 No.31
Vol.2016-EMB-40 No.31
2016/3/25

表 2: 単純手法と提案手法および提案手法（CSPF）の比較

			単純手法		提案手法（論理一致）			提案手法（CSPF）				
	PI	PO	ゲート数	UO 数 1	ゲート数	UO 数 2	時間（秒）	ゲート数	UO 数 3	時間（秒）	UO 比 1（%）	UO 比 2（%）
9symml	9	1	220	440	168	307	0.04	106	198	0.25	-55	-35.5
C880	60	26	413	826	304	569	0.2	284	560	191.22	-32.2	-1.58
C1908	33	25	501	1002	308	569	0.44	360	692	210.42	-30.94	+21.62
C3540	50	22	1089	2178	518	1000	2.82	504	992	365.95	-54.45	-0.8
C5315	178	123	1767	3534	1376	2663	3.59	856	1690	766.94	-52.18	-36.54
alu2	10	6	400	800	288	549	0.11	204	400	0.28	-50	-27.14
alu4	14	8	736	1472	599	1157	0.37	478	936	1.75	-36.41	-19.1
apex7	49	37	200	400	166	320	0.03	138	265	0.46	-33.75	-17.19
cht	47	36	148	296	121	215	0.01	84	159	0.15	-46.28	-26.05
dalu	75	16	1156	2312	447	879	0.63	140	275	5.77	-88.11	-68.71
example2	85	66	280	560	244	477	0.03	217	423	0.72	-24.46	-11.32
f51m	8	8	125	250	88	166	0.01	62	117	0.03	-53.2	-29.52
frg2	143	139	857	1714	761	1463	0.45	524	1028	40.93	-40.02	-29.73
i2	201	1	226	452	219	435	0.02	191	381	0.8	-15.71	-12.41
i6	138	67	510	1020	412	763	0.15	282	562	1.12	-44.9	-26.34
i7	199	67	630	1260	563	1065	0.2	294	587	2.25	-53.41	-44.88
i8	133	81	1305	2610	943	1840	0.9	861	1709	81.06	-34.52	-7.12
i10	257	224	2258	4516	1401	2672	5.03	1271	2475	2011.65	-45.19	-7.37
k2	45	45	1459	2918	383	741	1.31	308	591	2.27	-79.75	-20.24
pair	173	137	1451	2902	1192	2282	1.8	1078	2108	1923.82	-27.36	-7.62
rot	135	107	541	1082	421	809	0.55	390	746	1679.05	-31.05	-7.79
t481	16	1	1218	2436	1161	2303	0.26	607	1186	39.15	-51.31	-48.5
term1	34	10	213	426	176	334	0.02	102	196	1.28	-53.99	-41.32
too_large	38	3	511	1022	401	792	0.12	255	500	62.31	-51.08	-36.87
ttt2	24	21	176	352	131	255	0.03	112	219	0.24	-37.78	-14.12
vda	17	39	765	1530	187	347	0.26	142	269	0.56	-82.42	-22.48
x3	135	99	677	1354	590	1124	0.24	456	886	8.84	-34.56	-21.17
x4	94	71	352	704	312	591	0.06	273	536	2.72	-23.86	-9.31
合計	3303	1950	23298	46596	16040	30823	20.14	12474	24365	7468.97	-47.71	-20.95

スペースの関係上で本表に記載できなかった分の結果も含めた合計値である．なお，計算時間に関して，数値は小数第三位を四捨五入した数値であり，0.005 に満たない値は 0 と表記している．

実験結果より，提案手法（CSPF）は，単純手法と提案手法（論理一致）に比べてそれぞれ平均で約 47.71%，20.95%の UO 数を削減できた．提案手法（CSPF）では，未接続出力を削減する際に CSPF を用いていることにより，置き換え可能な未接続出力の候補が提案手法（論理一致）に比べて多い．それにより，ゲートの置き換えをより多くできたことが，提案手法（CSPF）が提案手法（論理一致）に比べて UO 数をより削減できた要因であると考えられる．

5. おわりに

本研究では，RQFP ゲートのみを用いて AQFP 論理回路を設計する手法について，出力論理の一致を利用した未接続出力の削減手法と CSPF を用いた未接続出力の削減手法の 2 つの手法を提案した．実験の結果，提案手法（CSPF）は，入力回路を AQFP 論理回路に変換するだけの単純手法や提案手法（論理一致）に比べ，それぞれ平均で約 47.71%，20.95%の未接続出力の数を削減できた．

本研究の提案手法を正確に評価するため，実際に物理的な回路を作成して消費電力を計測し，一般的な AQFP 論理回路と比較する必要がある．少なくとも，未接続出力 1 つあたりの増加消費電力を見積もり，その値から回路全体の消費電力を見積もった結果と一般的な AQFP 論理回路の消費電力を比較した評価を行う必要があると考えられる．

謝辞 本研究は JSPS 科研費 15H02679 の助成を受けたものです．

参考文献

[1] Ball, P.: Computer engineering: Feeling the heat, *Nature*, Vol. 492, pp. 174–176 (2012).
[2] Landauer, R.: Irreversibility and Heat Generation in the Computing Process, *IBM Journal of Research and Development*, Vol. 5, No. 3, pp. 183–191 (1961).
[3] Muroga, S., Kambayashi, Y., Lai, H. and Culliney, J.: The transduction method-design of logic networks based on permissible functions, *Computers, IEEE Transactions on*, Vol. 38, No. 10, pp. 1404–1424 (1989).
[4] Takeuchi, N., Ozawa, D., Yamanashi, Y. and Yoshikawa, N.: An adiabatic quantum flux parametron as an ultra-low-power logic device, *Superconductor Science and Technology*, Vol. 26, No. 3, p. 035010 (2013).
[5] Takeuchi, N., Yamanashi, Y. and Yoshikawa, N.: Reversible Computing Using Adiabatic Superconductor Logic, *Reversible Computation*, Lecture Notes in Computer Science, Vol. 8507, Springer International Publishing, pp. 21–23 (2014).
[6] YANG, S.: Logic synthesis and optimization benchmarks user guide version 3.0, *MCNC, 1991* (1991).