論　文

2×2-Joinを用いた二線式RSFQ論理回路の設計手法

小畑　幸嗣[†a)]　　高木　一義[†]　　高木　直史[†]

Design Method of Dual-Rail RSFQ Logic Circuits Using 2×2-Join

Koji OBATA[†a)], Kazuyoshi TAKAGI[†], and Naofumi TAKAGI[†]

あらまし　高速単一磁束量子（RSFQ）回路は，半導体回路では実現することが困難な，超高速計算を実現できるものと期待され，現在実用化に向けた研究が行われている．2×2-Joinは二線式RSFQ論理回路向きの基本論理回路であり，2×2-Joinを用いることにより，小面積で高速な回路を設計することができる．本論文では，2×2-Joinを基本論理回路に用いた，二線式RSFQ論理回路の設計手法を提案する．論理関数の表現のために，二分決定グラフ（BDD）をもとにした根共有型二分決定グラフ（RSBDD）を新たに導入する．与えられた論理関数からRSBDDを作成し，RSBDDから2×2-Joinを用いた二線式RSFQ論理回路を設計する．提案手法を実装し，いくつかのベンチマーク回路の設計に適用した結果，回路を高速に設計可能であった．

キーワード　2×2-Join，RSFQ論理回路，二線式論理，二分決定グラフ，設計手法

1. まえがき

超伝導体を用いた，高速単一磁束量子（Rapid Single Flux Quantum：RSFQ）回路[1]は，半導体回路では実現することが困難な，超高速計算を実現できるものと期待され，現在実用化に向けた研究が行われている．これまでのRSFQ回路の研究は，回路の動作実証を目的としたものが多く，設計される回路規模も小さなものであった．現在，製造技術の進歩により，大規模回路の製造，動作が可能になりつつあり，大規模回路の設計手法の開発が望まれている．

RSFQ論理回路では，パルス論理が用いられる．パルス論理では，1本の信号線で“0”，“1”の二つの論理値を表現することができない．そのため，論理値の表現法として，同期クロックを用いるクロック同期式[2]，2本の信号線を用いる二線式[2]が主に使用されている．クロック同期式は，厳密にタイミング設計を行うことができれば，高速な回路が設計可能である．一方，二線式は，同期クロックが必要ないため，回路のタイミング設計が容易になる．大規模回路の設計において，厳密なタイミング設計は煩雑な作業であり，タイミング設計の容易な設計手法の開発は重要である．そのため，二線式RSFQ論理回路の設計手法の研究が行われている[3]〜[5]．二線式では，一つの論理値の表現に2本の信号線を用いるため，クロック同期式に比べて配線の本数が増加し，それに伴って面積が増大することが多い．二線式RSFQ論理回路の研究では，配線の本数を減らし，面積を小さくすることが可能な基本論理回路，設計手法の開発が重要である．

本論文では，2×2-Join[4]を基本論理回路として用いた，二線式RSFQ論理回路の設計手法を提案する．2×2-Joinは，いくつかの合流回路（Confluence Buffer：CB）[1]，分岐回路（SPLitter：SPL）[1]と組み合わせることにより，様々な論理演算を実現することが可能である．提案手法では，論理関数の表現のために，二分決定グラフ（Binary Decision Diagram：BDD）[6], [7]をもとにした根共有型二分決定グラフ（Root Shared BDD：RSBDD）を新たに導入する．与えられた論理関数からRSBDDを作成し，RSBDDから2×2-Joinを用いた回路を設計する．提案する設計手法は，制御系の回路，データパス系の回路を設計する際の要素回路などを，系統的に高速に設計するためのものである．

本論文で提案する設計手法をC言語で実装し，いく

† 名古屋大学大学院情報科学研究科情報システム学専攻，名古屋市
Department of Information Engineering, Nagoya University, Furo-cho, Chikusa-ku, Nagoya-shi, 464-8603 Japan
a) E-mail : obata@takagi.nuie.nagoya-u.ac.jp

つかの回路の設計に適用したところ，17 入力 9 出力の回路（8 ビット加算器）等が数十ミリ秒で設計できた．本論文で提案した設計手法を用いることにより，2×2-Join を用いた，面積の小さな二線式 RSFQ 論理回路を設計することができる．

以下では，**2.** で 2×2-Join を用いた二線式 RSFQ 論理回路について述べ，**3.** で RSBDD を，**4.** で RSBDD を用いた回路の設計手法を提案する．

2. 2×2-Joinを用いた二線式RSFQ論理回路

二線式論理は，2 値の信号の伝搬に 2 本の信号線を用いる論理である．パルス論理の場合，“0” を表す信号線上にパルスが存在すれば論理値の “0” を，“1” を表す信号線上にパルスが存在すれば論理値の “1” を表す．両方の信号線上にパルスが存在しなければ，信号が存在しない状態を表す．二線式 RSFQ 論理回路では，回路中のすべての経路において，前のデータを追い越さないという制約のもと，次のデータを入力することができる．したがって，回路中のすべての経路の遅延時間をそろえることにより，高い動作周波数の回路を構成することが可能である．

2×2-Join は，二線式論理回路向きの基本論理回路であり，2×2-Join といくつかの合流回路（CB），分岐回路（SPL）を組み合わせることにより，回路を構成することができる．図 1 に，2×2-Join のジョセフソン回路図と入出力の関係，図 2 に，本論文中で使用する基本回路の表記を示す．2×2-Join は入力 A，B にそれぞれ二線式のデータが入力され，入力の組合せに応じて，4 個の出力（00，01，10，11）のうちただ一つにパルスを生じさせる．例えば，At と Bt にパルスが入力された場合は，11 からパルスが出力される．図 1(a) に示すように，2×2-Join は小さな基本論理回路であり，2×2-Join を用いることにより，二線式論理用の AND 回路や OR 回路などを用いる手法 [3] などと比べて，面積の小さな回路を設計することが可能である．

2×2-Join を用いた二線式 RSFQ 論理回路の例として，図 3 に全加算器（$S = X \oplus Y \oplus Z$，$C = X \cdot Y + Y \cdot Z + Z \cdot X$）を示す．図 3(a) は 2×2-Join の数が最小の回路 [4] である．一方，図 3(b) は 2×2-Join と CB を用いて AND や OR，XOR 回路を構成し，設計したものである．図 3(b) の回路では 2×2-Join が 5 個必要であるのに対し，図 3(a) では 2 個で構成される．2×2-Join を利用することで面積の小さな回路を設計することが可能となるが，そのような回路を人手で設計することは，回路規模が大きくなると困難である．本論文では，このような，2×2-Join を用いた面積の小さな回路を設計する，系統的な手法を提案する．

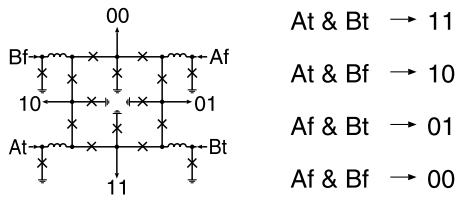


図 1　2×2-Join (a) 回路図，(b) 入出力の関係
Fig. 1　2×2-Join. (a) Schematic, (b) Input-output relations.

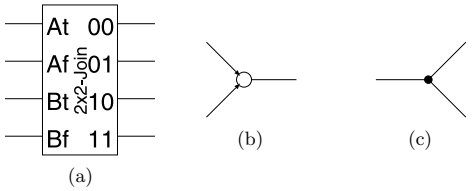


図 2　基本回路の表記 (a) 2×2-Join，(b) CB，(c) SPL
Fig. 2　Notations of basic circuits. (a) 2×2-Join, (b) CB, (c) SPL.

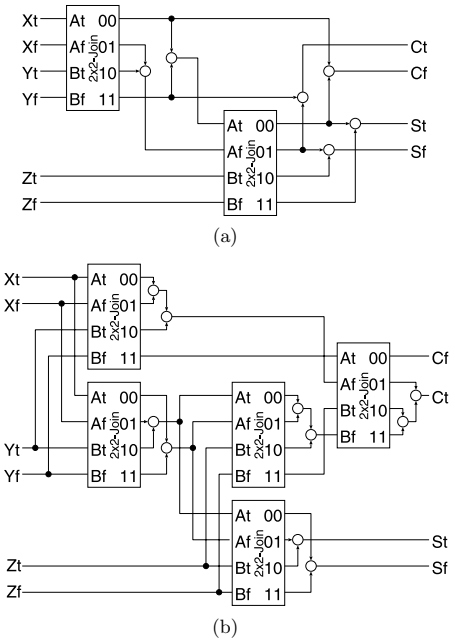


図 3　2×2-Join を用いた全加算器の二つの設計例
Fig. 3　Two design examples of full adder using 2×2-Joins.

3. 根共有型二分決定グラフ（RSBDD）

2×2-Join を用いた二線式 RSFQ 論理回路の設計を行うために，二分決定グラフ（BDD）に基づく根共有型二分決定グラフ（RSBDD）を新たに導入する．まず基本となる BDD について述べた後，RSBDD の定義，作成法，作成例について述べる．

3.1 二分決定グラフ（BDD）

二分決定グラフ（BDD）は，論理関数を表現する有向非巡回グラフである．節点集合は，変数をラベルとしてもつ非終端節点と，定数（0 または1）をラベルとしてもつ終端節点から構成される．枝集合は，各非終端節点から出る 2 本の枝から構成される．非終端節点は変数節点とも呼ばれ，変数節点の中には，入次数が 0 の根節点と呼ばれる節点が一つ存在する．終端節点は葉節点とも呼ばれる．非終端節点の出次数は 2 であり，終端節点の出次数は 0 である．非終端節点から出ている 2 本の枝は，それぞれラベルをもち，0 のラベルをもつ枝を 0 枝，1 のラベルをもつ枝を 1 枝と呼ぶ．

BDD は，論理関数のシャノン展開をグラフで表したものと考えることができる．変数に 0，1 の値割当が与えられると，BDD の根から順に枝のラベルの変数の値に従って，グラフをたどっていくことができる．変数の値が 0 の場合は 0 枝を，1 の場合は 1 枝をたどっていく．グラフをたどっていったときに，最終的に到達する終端節点の値が，その入力に対する論理関数の値となる．

BDD では，根節点から葉節点に至るすべての経路に現れる変数の順序を固定し，冗長な節点の削除と，等価な節点の統合を行うことによって，論理関数を一意に表現することが可能になる．冗長な節点とは，0 枝と 1 枝が同じ節点を指している節点である．また，二つの節点があり，変数名が同じで，0 枝と 1 枝がそれぞれ同じ節点を指しているとき，この二つの節点は等価である．このような BDD を既約順序付き二分決定グラフ [7] と呼ぶ．以降，本論文では BDD と表記した場合，この既約順序付き二分決定グラフを指す．BDD は EDA（Electronic Design Automation）の分野で広く用いられている．

BDD の例として，$X \oplus Y \oplus Z$ の BDD を図 4 に示す．本論文では，0 枝を破線で，1 枝を実線で表す．

3.2 根共有型二分決定グラフ（RSBDD）

根共有型二分決定グラフ（RSBDD）を次のように再帰的に定義する．すべての BDD の変数順序は同じであるとする．

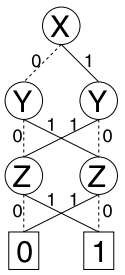


図 4 $X \oplus Y \oplus Z$ の BDD
Fig. 4 BDD of $X \oplus Y \oplus Z$.

（1） BDD は RSBDD である．

（2） 根節点の変数が同じ二つの RSBDD において，以下の節点の統合可能条件を満たす変数節点をすべて統合したものは RSBDD である．

[異なる RSBDD 間の変数節点の統合可能条件]

節点 u の親節点を u_{p_i}（$1 \leqq i \leqq m$），節点 v の親節点を v_{q_j}（$1 \leqq j \leqq n$）とする．m，n はそれぞれ節点 u，v の親節点の個数である．また，u と u_{p_i} 間の枝のラベル（0 枝ならば 0，1 枝ならば 1）を e_{p_i}，v と v_{q_j} 間の枝のラベルを e_{q_j} とする．節点 u と節点 v が統合可能であるとは，節点 u と節点 v の変数名が等しく，親節点とその間の枝のラベルの対の集合 $U_p = \{(u_{p_1}, e_{p_1}), \ldots, (u_{p_m}, e_{p_m})\}$，$V_q = \{(v_{q_1}, e_{q_1}), \ldots, (v_{q_n}, e_{q_n})\}$ に対して，$U_p = V_q$ が成り立つことである．ここで，$u_{p_i} = v_{q_j}$ とは，u_{p_i} と v_{q_j} が統合可能であるということである．

この条件を満たすとき，節点 u と節点 v を統合することができる．明らかに $m = n$ でなければならない．根節点の場合，親節点は存在しないので，変数名が同じであれば統合可能である．

二つの関数に対する BDD から RSBDD を作成する場合，その二つの BDD のすべての節点について，節点が統合可能かどうかを調べ，統合可能である節点をすべて統合する．三つ以上の BDD から作成する場合は，まず任意に二つの BDD を選択し RSBDD を作成した後，その作成した RSBDD と未処理の BDD から新たな RSBDD を作成すればよい．どのような順番で関数を選択しても，作成される RSBDD は同一である．根節点が異なる BDD が存在する場合，根節点を共有することができないため，複数個の RSBDD が作成される．関数が論理式で与えられた場合，BDD を構成しながら節点を統合していくことができる．節点や枝

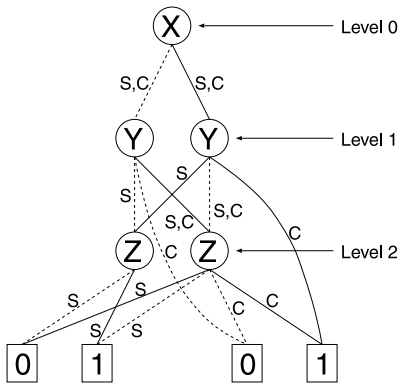


図 5　全加算器の RSBDD
Fig. 5　RSBDD of full adder.

がどの関数に属しているのかを識別するために，枝に関数名を付加する．

図 5 に全加算器（$S = X \oplus Y \oplus Z$，$C = X \cdot Y + Y \cdot Z + Z \cdot X$）の RSBDD を示す．関数 S のラベルの付いた枝と，その枝に接続している節点を抜き出すと，図 4 の BDD に一致する．本論文では，根節点から順に葉節点に向かって，レベル 0，レベル 1，... とする．また，RSBDD の節点数とは，葉を除いた総節点数とする．図 5 の RSBDD の場合，節点数は 5 である．

3.3　RSBDD の作成

複数の関数が論理式で与えられた場合の，RSBDD の作成法について述べる．論理式に従って RSBDD を作成していく．

まず関数を一つ選択し，その関数の論理式に従って BDD を作成する．以降一つずつ関数を選択し，BDD を作成していく．新たに作成される節点と，親節点の個数が変化した節点，親節点が他の節点と統合された節点については，これまでに作成した RSBDD の節点と統合できるかどうかを調べ，統合できるならば統合する．

ある二つの節点が統合可能かどうかを調べる場合，それらの節点の親をすべて調べる必要がある．ある節点の親の個数は，その節点に入力されている枝の本数に等しい．よって，ある BDD を考えたときの親の個数の総数は，その BDD の枝の総数に等しくなる．つまり，親の個数の総数は節点数の 2 倍となる．また，与えられた論理関数すべてに対して処理を行う必要がある．よって，RSBDD を作成する際の計算量は，BDD の節点数をすべての関数について合計した値に比例する．

3.4　RSBDD の作成例

図 6 に全加算器（$S = X \oplus Y \oplus Z$，$C = X \cdot Y + Y \cdot Z + Z \cdot X$）の RSBDD の作成過程を示す．変数順序は X，Y，Z とする．図では説明のため，節点に番号を付加してある．S，C の順で処理を行うものとする．

まずはじめに，S の BDD を作成する．図 6(a) が S の BDD である．続いて，$C = X \cdot Y + Y \cdot Z + Z \cdot X$ を前から順に処理していく．図 6(b) は，変数 X のみからなる BDD を作成した段階である．このとき，0 番の節点と 5 番の節点は互いに根節点であり，変数名が一致しているので統合することができる．統合したグラフを図 6 (c) に示す．実際には，図 6(b) のグラフは作成せず，図 6(c) のグラフを直接作成するが，ここでは説明のため図 6(b) のグラフも示しておく．

図 6(d) は，$X \cdot Y$ まで処理を行った段階である．2 番の節点と 6 番の節点は互いに変数名が等しい．また，親節点である 0 番の節点との間の枝のラベルも互いに 1 である．したがって，統合可能条件を満たし，統合することができる．統合したものが図 6(e) である．

図 6(f) は $X \cdot Y + Y \cdot Z$ まで処理を行った段階である．この段階では，1 番の節点と 7 番の節点が統合可能条件を満たすので統合することができ，統合したものが図 6(g) である．

図 6(h) が $C = X \cdot Y + Y \cdot Z + Z \cdot X$ まで処理を行った段階である．この段階では，4 番の節点と 8 番の節点を統合することができる．変数名は両者とも Z であり，親節点である 1 番節点との間の枝のラベルは 1，2 番節点との間の枝のラベルは 0 であるため，統合可能条件を満たす．統合したグラフが図 6(i) であり，図 5 の全加算器の RSBDD と一致する．図 6 (d)，(f)，(h) のグラフも図 6(b) のグラフと同様に実際には作成せず，図 6(e)，(g)，(i) のグラフを直接作成する．

4.　RSBDD を用いた回路の設計手法

3. で導入した RSBDD を用いた，二線式 RSFQ 論理回路の設計手法を提案する．

4.1　設 計 手 法

提案する設計手法では，与えられたすべての論理関数から RSBDD を作成し，その作成した RSBDD を用いて回路の設計を行う．複数個の RSBDD が作成される場合は，それぞれの RSBDD に対して処理を行う．RSBDD の根節点から葉節点に向かって回路を構成していく．根節点から葉節点に向かって信号が流れる回路を構成する．根節点とレベル 1 の節点から回路の 1

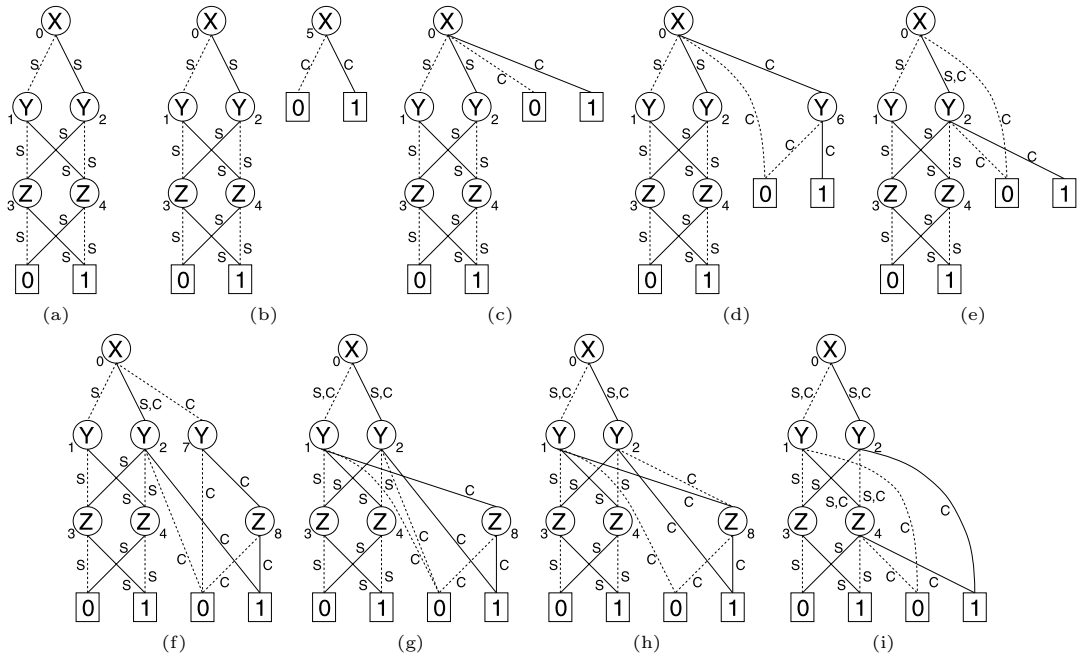


図 6　全加算器の RSBDD の作成過程
Fig. 6　Making process of full adder's RSBDD.

段目の 2×2-Join を構成し，順次レベル i （$i \geqq 2$）の節点から i 段目の 2×2-Join を構成する．最後に葉から回路の出力を構成する．

2×2-Join の 4 個の出力（00，01，10，11）から適切な信号を選択するために，RSBDD の節点に 0/1 の出力を選択するための番号付けを行う．この番号を，出力選択番号と呼ぶ．出力選択番号が 0 の場合，対応する 2×2-Join の出力の 00，01 のどちらか，1 の場合，10，11 のどちらかを，対応する節点から出ている枝のラベルに応じて選択する．

根節点とレベル 1 の節点（最大二つ）から 1 段目の 2×2-Join を構成する．2×2-Join の入力 At に根節点の変数の $true$ 信号，Af に根節点の変数の $false$ 信号，Bt にレベル 1 の節点の変数の $true$ 信号，Bf にレベル 1 の節点の変数の $false$ 信号を接続する．根節点，処理を行ったレベル 1 の節点を処理済みとする．レベル 1 の節点の出力選択番号は，根の 0 枝が入力されている場合は 0，1 枝が入力されている場合は 1 とする．

レベル 2 以降の節点から 2 段目以降の 2×2-Join を構成する．レベルの小さな節点から順次処理を行っていく．レベル i の節点から i 段目の 2×2-Join を構成する際には，2×2-Join の入力 Bt にレベル i の節点の変数の $true$ 信号，Bf に $false$ 信号，At，Af にはそれぞれ，RSBDD から求まる前段までの回路出力を接続する．処理を行った節点は出力選択番号を 1 とし，処理済みとする．

At には，処理中の節点に入力されている枝に対応する信号線を接続する．処理中の節点の親が根節点の場合は，その親節点との間の枝が 0 枝の場合，根節点の変数の $false$ 信号，1 枝の場合は $true$ 信号が求める信号となる．根節点以外の場合は，その親節点の出力選択番号と枝のラベル（0/1）の連接（00，01，10，11）に一致する，親節点に対応する 2×2-Join の出力信号が求める信号となる．それらの信号をすべての親について求め，CB で一つの信号線に集約し At に接続する．

一方，Af には，処理中の節点に入力されている枝を，根まで逆にすべての経路をたどっていったときに通る節点の枝で，経路に含まれていないものに対応する信号線を接続する．対応する信号線の求め方は At の場合と同様である．ここで求めた Af に接続する信号線と，同じレベルの未処理節点の中で，At に接続する信号線を求めた場合，両者が一致するものがあるか調べ，そのような未処理節点が存在すれば，その未処理節点の出力選択番号を 0 とし，処理済みとする．このような節点が存在する場合，二つの節点で一つの 2×2-Join を共有することが可能である．

すべての未処理節点に対して，このような処理を行う．

回路出力は，0 の葉から $false$ 信号が，1 の葉から $true$ 信号が構成される．葉に入力されている枝に対応する信号線を At の場合と同様に求め，CB で一つの信号に集約する．

回路中の 2×2-Join の段数は，RSBDD の変数の個数から 1 を引いたものに等しくなり，2×2-Join の個数は，レベル 1 の節点を除いた節点数以下となる．どれだけの個数の 2×2-Join で回路を構成可能かは，いくつの節点で 2×2-Join を共有可能かで決定される．中間変数を導入して RSBDD を分解することにより，回路段数を低減することができる．

4.2 回路の設計例

提案手法による回路の設計例として，全加算器の設計を考える．全加算器の RSBDD は図 6(i) である．全加算器の設計過程を図 7 に示す．節点の番号は図 6(i) 中の節点番号である．

まず，根節点とレベル 1 の二つの節点から 1 段目の 2×2-Join を構成する．2×2-Join を一つ用意し，入力 At に根の変数の $true$ 信号（Xt），Af に $false$ 信号（Xf），Bt にレベル 1 の変数の $true$ 信号（Yt），Bf に $false$ 信号（Yf）を接続する．根節点とレベル 1 の二つの節点を処理済みとする．根節点の 0 枝が入力されているレベル 1 の節点（1 番の節点）の出力選択番号を 0 とし，1 枝が入力されている節点（2 番の節点）の出力選択番号を 1 とする．この段階まで設計した回路が図 7(a) である．

続いて，レベル 2 の各節点から 2 段目の 2×2-Join を構成する．まず，左側の節点（3 番の節点）から処理を行う．新たな 2×2-Join を用意し，入力 Bt にレベル 2 の変数の $true$ 信号（Zt），Bf に $false$ 信号（Zf）を接続する．この節点を処理済みとし，出力選択番号を 1 とする．At には，この節点に入力されている枝（1 番の節点の 0 枝，2 番の節点の 1 枝）に対応する信号線を接続する．1 番の節点の出力選択番号は 0 であり，2 番の節点の出力選択番号は 1 であるので，1 段目の 2×2-Join の出力 00 と 11 が対応する信号線となる．したがって，この二つの信号線を CB を用いて一つに集約し接続する．ここまでで図 7(b) の回路が得られる．

Af には，入力されている枝を根までたどっていったときに通る節点の枝で，経路に含まれていない枝（1 番の節点の 1 枝，2 番の節点の 0 枝）に対応する信号線を接続する．したがって，1 段目の 2×2-Join の出力 01 と 10 を CB で一つの信号線に集約し，接続する．この Af に接続される信号線は，レベル 2 の未処理節点である，4 番の節点に入力されている枝に対応する信号線と同一である．そのため，この 4 番の節点の出力選択番号を 0 とし，処理済みとすることができる．これでレベル 2 の節点の処理は終了である．回路図を図 7(c) に示す．

葉についても，入力されている枝に対応する信号線を求め，回路出力を構成する．関数 S の回路出力（St, Sf）を構成すると，図 7(d) の回路が得られる．

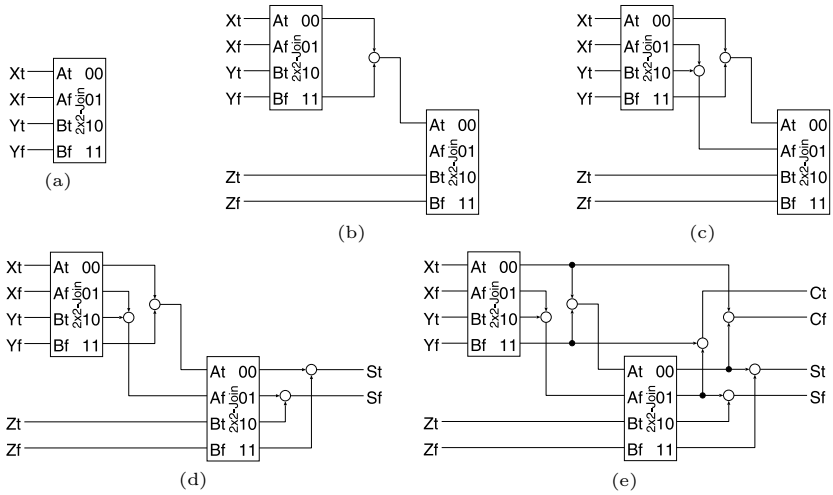


図 7　全加算器の設計過程
Fig. 7　Design process of full adder.

関数 C についても同様に回路出力（Ct，Cf）を構成すると，全加算器が得られる．回路図を図 7(e) に示す．

全加算器の RSBDD（図 6(i)）において，レベル 1 の節点を除いた節点数は 3 であり，レベル 2 の変数 Z の二つの節点で一つの 2×2-Join を共有している．したがって，全加算器の回路中の 2×2-Join の数は 2 個である．2×2-Join の段数も，全加算器は 3 変数であるため，2 段である．

4.3 設計手法の計算量

提案手法の計算量は，Af に接続する信号線を求める部分が最大となる．各節点について，Af に接続する信号線を求めるとき，その節点の祖先すべてを調べる必要がある．しかし，各節点を処理する際に，その節点までの処理結果，つまり，経路に含まれる枝と，含まれない枝のそれぞれを記憶しておけば，節点の親のみを調べることによって Af に接続する信号線を求めることができる．節点の親の総数は，枝の総数に等しい．また，各節点を処理する際に保存する枝の本数は，節点一つにつき枝の総数を超えることはない．これらのことより，ある節点の Af に接続する信号線を求める際の計算量は，RSBDD の枝の総数とその処理を行っている節点の親の個数の積に比例し，設計手法全体の計算量は RSBDD の枝の総数の 2 乗に比例する．

4.4 設計手法の評価

提案手法の評価を行うために，提案手法を C 言語で実装した．実験環境は Athlon 1.2 GHz，メインメモリ 256MB の Linux マシンである．加算器，比較器，いくつかのベンチマーク回路 [9] の設計に提案手法を適用した結果，得られた RSBDD の節点数と 2×2-Join の個数を表 1 に示す．add1 から add8 が加算器，comp1 から comp8 が比較器，5xp1 以下がベンチマーク回路である．表 1 の回路に対する実行は，すべて瞬時（数十ミリ秒）に終了した．add4，add8，comp4，comp8 は，add1 や comp1 を 4 段若しくは 8 段直列に接続した回路ではなく，異なった回路が生成されている．17 入力，9 出力の回路（add8）も瞬時に設計することが可能であった．

文献 [5] においても複数の BDD を統合して，節点を一つずつ Bina [5] で置き換えることにより設計を行う手法が示されているが，統合の詳細については示されていない．そこで，本論文で示した RSBDD を作成することにより，BDD の統合を行い，節点を一つずつ Bina で置き換える回路と，2×2-Join を用いた回路を比較する．表 2 に CONNECT セルライブラリ [8] に収録されている 2×2-Join セルと Bina セルのジョセフソン接合数，遅延時間を示す．本論文で提案している 2×2-Join を用いる手法では，最大で RSBDD の二つの節点に対して，一つの 2×2-Join を割り当てることが可能である．これらのことより，提案手法は，Bina を割り当てる手法で必要となる Bina の個数よりも，少ない個数の 2×2-Join で回路を設計することができる．最良の場合で Bina を割り当てる手法の半分程度の面積，最悪でも同等の面積である．実際，加算器は Bina を用いる場合と比べて，半分の個数の 2×2-Join で回路を構成可能である．基本論理回路の段数は，どちらの手法も同じであるので，提案手法は遅延時間に関しても同等以上の性能をもつ．

表 1　RSBDD の節点数と 2×2-Join の個数
Table 1　Number of RSBDD's nodes and 2×2-Joins.

回路名	入力	出力	RSBDD の節点数	2×2-Join の個数
add1	3	2	5	2
add4	9	5	17	8
add8	17	9	33	16
comp1	3	1	4	2
comp4	9	1	13	8
comp8	17	1	25	16
5xp1	7	10	46	37
9sym	9	1	33	31
apex4	9	19	383	381
bw	5	28	45	43
clip	9	5	56	53
con1	7	2	13	11
misex1	8	7	18	16
rd53	5	3	17	12
rd73	7	3	29	22
rd84	8	4	43	35
xor5	5	1	9	4

表 2　2×2-Join と Bina のジョセフソン接合数，遅延時間（NEC 2.5kA/cm^{2}Nb プロセス，2.5 mV バイアス）
Table 2　Josephson junctions and delay time of 2×2-Join and Bina (NEC 2.5kA/cm^2 Nb process, 2.5 mV bias voltage).

	ジョセフソン接合数	遅延時間（平均値，ps）
2×2-Join	20	18.875
Bina	27	24.6

5. むすび

本論文では，2×2-Join を用いた二線式 RSFQ 論理回路の設計を行うための，設計手法について述べた．根を共有した BDD である，RSBDD を新たに導入した．提案手法をいくつかの回路の設計に適用したところ，高速に，2×2-Join を用いた二線式 RSFQ 回路が設計可能であった．このことより，大規模回路の設計にも適用することができると考えられる．本論文で提案した設計手法を用いることにより，2×2-Join を用いた，面積の小さい二線式 RSFQ 論理回路の，系統的な設計が可能になる．

本論文で提案した設計手法では，2×2-Join の段数は与えられた論理関数の変数の個数から 1 を引いた値になる．与えられた論理関数の中に，並列に計算できる部分が存在する場合には，設計される回路の段数がフルカスタムで設計した場合よりも深くなる．関数の分解を行うことによって，並列性を抽出することができれば，回路の段数を低減することができる．関数の分解を行い，RSBDD の最適な変数順序を求めることにより，フルカスタムで設計した二線式 RSFQ 論理回路と同等以上の性能の回路が設計できると考えられる．このような RSBDD における最適な変数の順序付け，関数の分解については今後の課題である．また，本論文で提案した設計手法では，回路の配線については考慮していない．配線の考慮についても今後の課題である．

謝辞　本研究は，低消費電力型超電導ネットワークデバイスの開発の研究として，ISTEC を通じて，新エネルギー・産業技術総合開発機構の委託により実施したものである．御討論頂いた本プロジェクト関係者の皆様に感謝致します．

文　献

[1] K.K. Likharev and V.K. Semenov, “RSFQ logic / memory family: A new Josephson-junction technology for sub-terahertz-clock-frequency digital systems,” IEEE Trans. Appl. Supercond., vol.1, no.1, pp.3–28, March 1991.

[2] K. Gaj, E.G. Friedman, and M.J. Feldman, “Timing of multi-gigahertz rapid single flux quantum digital circuits,” J. VLSI Signal Process., vol.16, no.2/3, pp.247–276, June/July 1997.

[3] M. Maezawa, I. Kurosawa, M. Aoyagi, H. Nakagawa, Y. Kameda, and T. Nanya, “Rapid single-flux-quantum dual-rail logic for asynchronous circuits,” IEEE Trans. Appl. Supercond., vol.7, no.2, pp.2705–2708, June 1997.

[4] Y. Kameda, S. Polonsky, M. Maezawa, and T. Nanya, “Primitive-level pipelining method on delay-insensitive model for RSFQ pulse-driven logic,” Proc. 4th International Symposium on Advanced Research in Asynchronous Circuits and Systems, pp.262–273, San Diego, CA, March 1998.

[5] N. Yoshikawa and J. Koshiyama, “Top-down RSFQ logic design based on a binary decision diagram,” IEEE Trans. Appl. Supercond., vol.11, no.1, pp.1098–1101, March 2001.

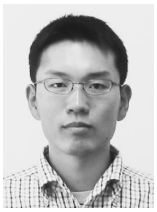
[6] S.B. Akers, “Binary decision diagrams,” IEEE Trans. Comput., vol.C-27, no.6, pp.509–516, June 1978.

[7] R.E. Bryant, “Graph-based algorithm for boolean function manipulation,” IEEE Trans. Comput., vol.C-35, no.8, pp.677–691, Aug. 1986.

[8] S. Yorozu, Y. Kameda, H. Terai, A. Fujimaki, T. Yamada, and S. Tahara, “A single quantum standard logic cell library,” Physica C, vol.378-381, pp.1471–1474, Oct. 2002.

[9] R. Lisanke, “Logic synthesis and optimization benchmarks user guide ver.2.0,” Technical report, Microelectronics Center of North Carolina, Dec. 1988.

（平成 16 年 8 月 27 日受付，10 月 22 日再受付）



小畑　幸嗣（学生員）

平 15 名大・工・電気電子情報工卒．現在同大大学院修士課程在学中．単一磁束量子回路の設計手法に関する研究に従事．

高木　一義（正員）

平 3 京大・工・情報工卒．平 5 同大大学院修士課程了．平 7 奈良先端大助手．平 11 名大・工・情報工助手．平 12 同講師．工博．現在，計算理論，VLSI 設計，VLSI CAD アルゴリズムの研究に従事．

高木　直史（正員）

昭 56 京大・工・情報工卒，昭 58 同大大学院修士課程了，昭 63 京大工博．昭 59 京大・工・情報工助手，平 3 同助教授，平 6 名大・工・情報工助教授，平 10 同教授，平 15 名大・情報科学・教授．算術演算回路，ハードウェアアルゴリズム，論理設計等の研究に従事．平 7 日本 IBM 科学賞，情報処理学会坂井記念特別賞等受賞．