(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2005-182654
(P2005-182654A)
(43)公開日　平成17年7月7日(2005.7.7)

(51)Int.Cl.[7]	F I		テーマコード（参考）
G06F 7/00	G06F 7/00	E	5B022
G06F 17/50	G06F 17/50	654M	5B046
H01L 21/82	G06F 17/50	656A	5F064
// H03K 19/173	H01L 21/82	A	5J042
	H01L 21/82	C	

審査請求 未請求　請求項の数 22　ＯＬ　（全 36 頁）　最終頁に続く

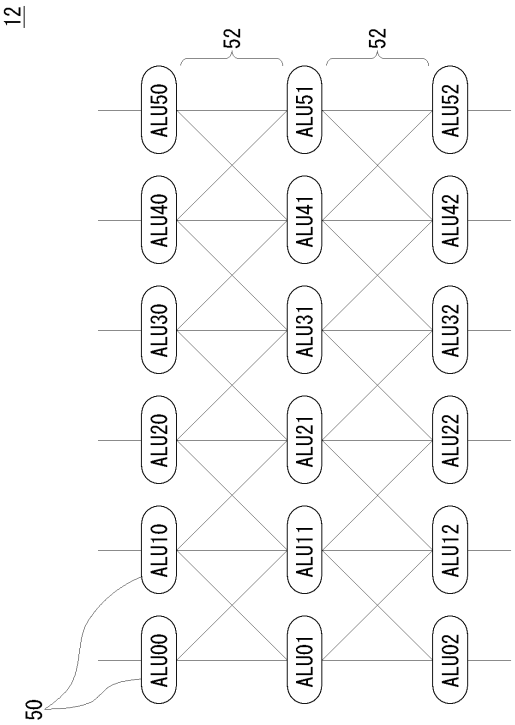
(21)出願番号　特願2003-425657 (P2003-425657)
(22)出願日　平成15年12月22日 (2003.12.22)

(71)出願人　000001889
三洋電機株式会社
大阪府守口市京阪本通２丁目５番５号
(74)代理人　100105924
弁理士　森下　賢樹
(72)発明者　岡田　誠
大阪府守口市京阪本通２丁目５番５号　三洋電機株式会社内
(72)発明者　平松　達夫
大阪府守口市京阪本通２丁目５番５号　三洋電機株式会社内
(72)発明者　中島　洋
大阪府守口市京阪本通２丁目５番５号　三洋電機株式会社内

最終頁に続く

(54)【発明の名称】リコンフィギュラブル回路、リコンフィギュラブル回路を備えた処理装置、リコンフィギュラブル回路における論理回路の機能決定方法、回路生成方法および回路

(57)【要約】
【課題】　回路規模を削減したリコンフィギュラブル回路を提供する。
【解決手段】　本発明のリコンフィギュラブル回路１２は、機能の変更が可能な複数のＡＬＵ５０を備える。複数のＡＬＵ５０はマトリックス状に配置され、ＡＬＵの各段の間には、ＡＬＵの接続を選択的に確立可能な少なくとも１つの接続部５２が設けられている。この接続部５２は、上下段における全ての論理回路同士を接続可能とするものではなく、論理回路が、別の段に属する一部の論理回路のみと接続可能であるように構成される。接続を制限することにより、回路規模を削減することができる。
【選択図】　図８



【特許請求の範囲】
【請求項１】
機能の変更が可能な論理回路の複数の集合体と、
それぞれの集合体の間に設けられて、集合体間の論理回路の接続を選択的に確立可能な少なくとも１つの接続部とを備えたリコンフィギュラブル回路であって、
接続部は、一方の集合体における論理回路が、別の集合体に含まれる一部の論理回路のみと接続可能であるように構成されることを特徴とするリコンフィギュラブル回路。
【請求項２】
接続部は、集合体間において、物理的に近接して配置された論理回路同士を接続可能とするように構成されることを特徴とする請求項１に記載のリコンフィギュラブル回路。
【請求項３】
複数の論理回路がマトリックス状に配列され、各段に配列された複数の論理回路が、論理回路の集合体を構成して、前段の集合体における処理結果が接続部において選択的に確立される接続にしたがって後段の集合体に引き渡され、
前段の論理回路の出力に接続可能な後段の集合体における論理回路は、同列に配置された論理回路と、その左右に配置された論理回路に制限されることを特徴とする請求項１または２に記載のリコンフィギュラブル回路。
【請求項４】
論理回路はＡＬＵであることを特徴とする請求項１から３のいずれかに記載のリコンフィギュラブル回路。
【請求項５】
機能の変更が可能なリコンフィギュラブル回路と、
前記リコンフィギュラブル回路に所期の回路を構成するための構成情報を供給する設定部と、
前記リコンフィギュラブル回路に、同時実行可能な複数の回路の構成情報を同時に供給するように前記設定部を制御する制御部と、
を備えることを特徴とする処理装置。
【請求項６】
前記リコンフィギュラブル回路の出力を、前記リコンフィギュラブル回路の入力に接続する経路部をさらに備え、
前記制御部は、複数のサイクルで、複数の構成情報を前記リコンフィギュラブル回路に順次供給するように前記設定部を制御し、ある構成情報により前記リコンフィギュラブル回路上で構成された回路の出力を、前記経路部を通じて次の構成情報により構成される回路の入力に供給させることを特徴とする請求項５に記載の処理装置。
【請求項７】
リコンフィギュラブル回路における論理回路の集合体の間の接続制限に応じて、入力データから出力データに至るデータの流れを表現するデータフローグラフを生成するＤＦＧ生成部と、
生成されたデータフローグラフをもとに、構成情報を生成する構成情報生成部と、
をさらに備えることを特徴とする請求項５または６に記載の処理装置。
【請求項８】
論理回路の機能をノードとして表現して、入力データから出力データに至るデータの流れをノードの接続で表現するデータフローグラフを生成するＤＦＧ生成部と、
生成されたデータフローグラフをもとに、構成情報を生成する構成情報生成部と、
機能の変更が可能なリコンフィギュラブル回路と、
前記リコンフィギュラブル回路に所期の回路を構成するための構成情報を供給する設定部とを備えた処理装置であって、
前記ＤＦＧ生成部は、あるノードの出力数が、対応する論理回路が接続できる論理回路の数よりも多い場合は、データフローグラフにおいて、入力を同一とする別のノードを複製して、ノード出力を分散させることを特徴とする処理装置。

【請求項９】
論理回路の機能をノードとして表現して、入力データから出力データに至るデータの流れをノードの接続で表現するデータフローグラフを生成するＤＦＧ生成部と、
生成されたデータフローグラフをもとに、構成情報を生成する構成情報生成部と、
機能の変更が可能なリコンフィギュラブル回路と、
前記リコンフィギュラブル回路に所期の回路を構成するための構成情報を供給する設定部とを備えた処理装置であって、
前記ＤＦＧ生成部は、あるノードの出力数が、対応する論理回路が接続できる論理回路の数よりも多い場合は、データフローグラフにおいて、処理を行わないスルーノードを、前記ノードの出力ノードの一部とともに、前記ノードの出力と接続するように追加し、スルーノードの後に残りの出力ノードを配置することで、ノード出力を分散させることを特徴とする処理装置。
【請求項１０】
機能の変更が可能な論理回路の複数の集合体と、それぞれの集合体の間に設けられて、論理回路の接続を所定の制限下で選択的に確立可能な少なくとも１つの接続部とを備えたリコンフィギュラブル回路における論理回路の機能を決定する方法であって、
論理回路の機能をノードとして表現して、入力データから出力データに至るデータの流れをノードの接続で表現するデータフローグラフを生成するＤＦＧ化処理ステップと、
データフローグラフの各ノードを、リコンフィギュラブル回路の論理回路に対応付けるマッピング処理ステップと、
を備えることを特徴とする機能決定方法。
【請求項１１】
前記ＤＦＧ化処理ステップは、
同時実行可能な複数のサブデータフローグラフを合成するステップを含むことを特徴とする請求項１０に記載の機能決定方法。
【請求項１２】
前記合成ステップは、
第１のサブデータフローグラフを、第１のサブデータフローグラフに入力データを直接的または間接的に供給せず、且つ、第１のサブデータフローグラフからの出力データを直接的または間接的に必要としない第２のサブデータフローグラフと合成することを特徴とする請求項１１に記載の機能決定方法。
【請求項１３】
前記マッピング処理ステップは、
データフローグラフにおける入力端に近い第１ノードを第１論理回路に対応付けるステップと、
データフローグラフにおいて前記第１ノードの出力の１つに接続された第２ノードを、リコンフィギュラブル回路において第１論理回路と接続可能な第２論理回路に対応付けるステップと、
を含むことを特徴とする請求項１０から１２のいずれかに記載の機能決定方法。
【請求項１４】
前記マッピング処理ステップは、
データフローグラフにおいて前記第２ノードの入力のうち前記第１ノードとは異なる第３ノードを、リコンフィギュラブル回路において第２論理回路の入力に接続可能であり、かつ第１論理回路に対応付けられていない第３論理回路に対応付けるステップを含むことを特徴とする請求項１３に記載の機能決定方法。
【請求項１５】
前記マッピング処理ステップは、
データフローグラフにおいて前記第２ノードの入力のうち前記第１ノードとは異なる第３ノードを、リコンフィギュラブル回路において第２論理回路の入力に接続可能であり、かつ第１論理回路に対応付けられていない論理回路が存在しない場合には、前記第２ノー

ドを、前記第１論理回路の出力と接続可能であり且つ前記第２論理回路とは異なる第４論理回路に対応付けるステップを含むことを特徴とする請求項１３に記載の機能決定方法。
【請求項１６】
前記マッピング処理ステップは、
第３ノードを、前記第４論理回路の入力と接続可能であり、且つ、未だ対応付けがなされていない第５論理回路に対応付けるステップを含むことを特徴とする請求項１５に記載の機能決定方法。
【請求項１７】
前記ＤＦＧ化処理ステップは、
データフローグラフの段数が、リコンフィギュラブル回路の集合体の個数を超える場合は、データフローグラフを、段数が集合体の個数より少ない複数のサブデータフローグラフに分割することを特徴とする請求項１０から１２のいずれかに記載の機能決定方法。
【請求項１８】
前記ＤＦＧ化処理ステップは、
データフローグラフの列数が、リコンフィギュラブル回路の集合体における論理回路の個数を超える場合は、データフローグラフを、列数が集合体における論理回路の個数より少ない複数のサブデータフローグラフに分割することを特徴とする請求項１０から１２のいずれかに記載の機能決定方法。
【請求項１９】
前記ＤＦＧ化処理ステップは、
データフローグラフにおいて、あるノードの出力数が、対応する論理回路が接続できる論理回路の数よりも多い場合に、そのデータフローグラフを、そのノードの出力のうちの第１出力を出力としてもつ第１のサブデータフローグラフと、第１出力に接続される入力をもつ第２のサブデータフローグラフに分割することを特徴とする請求項１０から１２のいずれかに記載の機能決定方法。
【請求項２０】
前記マッピング処理ステップは、
データフローグラフにおけるノードの全てをリコンフィギュラブル回路における論理回路に対応付けられない場合に、対応付けられなかったノードを含むデータフローグラフに対して、再度マッピング処理を行うことを特徴とする請求項１０から１２のいずれかに記載の機能決定方法。
【請求項２１】
複数の回路素子の機能および配置を決定して、所期の回路を生成する方法であって、
回路素子の機能をノードとして表現して、入力データから出力データに至るデータの流れをノードの接続で表現するデータフローグラフを生成するＤＦＧ化処理ステップと、
データフローグラフの各ノードを、生成するべき回路の回路素子の配置位置に対応付けるマッピング処理ステップと、
を備えることを特徴とする回路生成方法。
【請求項２２】
回路素子の機能をノードとして表現して入力データから出力データに至るデータの流れをノードの接続で表現するデータフローグラフを生成し、データフローグラフの各ノードを、生成するべき回路の回路素子の配置位置に対応付けて、その配置位置にノードの示す回路素子を形成することで生成された回路。
【発明の詳細な説明】
【技術分野】
【０００１】
この発明は、集積回路技術に関し、特にリコンフィギュラブル回路を備えた処理装置などに関する。
【背景技術】
【０００２】

近年、アプリケーションに応じてハードウェアの動作を変更可能なリコンフィギュラブルプロセッサの開発が進められている。リコンフィギュラブルプロセッサを実現するためのアーキテクチャとしては、ＤＳＰ(Digital Signal Processor)や、ＦＰＧＡ(Field Programmable Gate Array)を用いる方法が存在する。

【０００３】

ＦＰＧＡ（Field Programmable Gate Array）はＬＳＩ製造後に回路データを書き込んで比較的自由に回路構成を設計することが可能であり、専用ハードウエアの設計に利用されている。ＦＰＧＡは、論理回路の真理値表を格納するためのルックアップテーブル（ＬＵＴ）と出力用のフリップフロップからなる基本セルと、その基本セル間を結ぶプログラマブルな配線リソースとを含む。ＦＰＧＡでは、ＬＵＴに格納するデータと配線データを書き込むことで目的とする論理演算を実現できる。しかし、ＦＰＧＡでＬＳＩを設計した場合、ＡＳＩＣ（Application Specific IC）による設計と比べると、実装面積が非常に大きくなり、コスト高になる。そこで、ＦＰＧＡを動的に再構成することで、回路構成の再利用を図る方法が提案されている（例えば、特許文献１参照。）。

【特許文献１】特開平１０－２５６３８３号公報

【発明の開示】

【発明が解決しようとする課題】

【０００４】

ＦＰＧＡは回路構成の設計自由度が高く、汎用的である反面、全ての基本セル間の接続を可能とするため、多数のスイッチとスイッチのＯＮ／ＯＦＦを制御するための制御回路を含む必要があり、必然的に制御回路の実装面積が大きくなる。また、基本セル間の接続に複雑な配線パターンをとるため、配線が長くなる傾向があり、さらに１本の配線に多くのスイッチが接続される構造のため、遅延が大きくなる。そのため、ＦＰＧＡによるＬＳＩは、試作や実験のために利用されるにとどまることが多く、実装効率、性能、コストなどを考えると、量産には適していない。さらに、ＦＰＧＡでは、多数のＬＵＴ方式の基本セルに構成情報を送る必要があるため、回路のコンフィグレーションにはかなりの時間がかかる。そのため、瞬時に回路構成の切り替えが必要な用途にはＦＰＧＡは適していない。

【０００５】

それらの課題を解決するため、近年、ＡＬＵ(Arithmetic Logic Unit)と呼ばれる基本演算機能を持つ多機能素子を多段に並べたＡＬＵアレイの検討が行われるようになった。ＡＬＵアレイでは、処理が上から下の一方向に流れるので、水平方向の配線は不要である。なお、ＦＰＧＡのように、１つのＡＬＵから全てのＡＬＵに接続することもできるが、この場合、データをどのＡＬＵに渡すこともできる代わりに、配線数および接続スイッチの数が膨大となるため、回路規模が増大する原因となる。

【０００６】

本発明はこうした状況に鑑みてなされたもので、その目的は、回路規模の縮小化に貢献するリコンフィギュラブル回路およびその周辺技術の提供にある。

【課題を解決するための手段】

【０００７】

上記課題を解決するために、本発明のある態様は、機能の変更が可能な論理回路の複数の集合体と、それぞれの集合体の間に設けられて、集合体間の論理回路の接続を選択的に確立可能な少なくとも１つの接続部とを備えたリコンフィギュラブル回路を提供する。このリコンフィギュラブル回路において、接続部は、一方の集合体における論理回路が、別の集合体に含まれる一部の論理回路のみと接続可能であるように構成される。このように、接続部において論理回路間の接続制限をかけることにより、ハードウェア的に接続部における配線数を低減して、スイッチ等の構成も低減することができるため、回路規模を小さくすることが可能となり、省電力化が図れるとともに、経済的にも有利となる。

【０００８】

本発明の別の態様は、機能の変更が可能なリコンフィギュラブル回路と、前記リコンフ

ィギュラブル回路に所期の回路を構成するための構成情報を供給する設定部と、前記リコンフィギュラブル回路に、同時実行可能な複数の回路構成情報を同時に供給するように前記設定部を制御する制御部とを備える処理装置を提供する。この処理装置によると、並列処理可能な回路をリコンフィギュラブル回路上に同時に形成することができるため、処理時間の短縮化を図ることが可能となる。

【０００９】

本発明のさらに別の態様は、論理回路の機能をノードとして表現して、入力データから出力データに至るデータの流れをノードの接続で表現するデータフローグラフを生成するＤＦＧ生成部と、生成されたデータフローグラフをもとに、構成情報を生成する構成情報生成部と、機能の変更が可能なリコンフィギュラブル回路と、前記リコンフィギュラブル回路に所期の回路を構成するための構成情報を供給する設定部とを備えた処理装置を提供する。この処理装置において、前記ＤＦＧ生成部は、あるノードの出力数が、対応する論理回路が接続できる論理回路の数よりも多い場合は、データフローグラフにおいて、入力を同一とする別のノードを複製して、ノード出力を分散させてもよい。これにより、接続が制限されるリコンフィギュラブル回路において、接続数が多いために直接的にノードをマッピングすることができない場合であっても、そのノードを複製してノード出力を分散させることで、所期の回路と等価な回路をリコンフィギュラブル回路上に構成することが可能となる。

【００１０】

本発明のさらに別の態様は、論理回路の機能をノードとして表現して、入力データから出力データに至るデータの流れをノードの接続で表現するデータフローグラフを生成するＤＦＧ生成部と、生成されたデータフローグラフをもとに、構成情報を生成する構成情報生成部と、機能の変更が可能なリコンフィギュラブル回路と、前記リコンフィギュラブル回路に所期の回路を構成するための構成情報を供給する設定部とを備えた処理装置を提供する。この処理装置において、前記ＤＦＧ生成部は、あるノードの出力数が、対応する論理回路が接続できる論理回路の数よりも多い場合は、データフローグラフにおいて、処理を行わないスルーノードを、前記ノードの出力ノードの一部とともに、前記ノードの出力と接続するように追加し、スルーノードの後に残りの出力ノードを配置することで、ノード出力を分散させてもよい。これにより、接続が制限されるリコンフィギュラブル回路において、接続数が多いために直接的にノードをマッピングすることができない場合であっても、スルーノードを配置してノード出力を分散させることで、所期の回路と等価な回路をリコンフィギュラブル回路上に構成することが可能となる。

【００１１】

本発明のさらに別の態様は、機能の変更が可能な論理回路の複数の集合体と、それぞれの集合体の間に設けられて、論理回路の接続を所定の制限下で選択的に確立可能な少なくとも１つの接続部とを備えたリコンフィギュラブル回路における論理回路の機能を決定する方法を提供する。この機能決定方法は、論理回路の機能をノードとして表現して、入力データから出力データに至るデータの流れをノードの接続で表現するデータフローグラフを生成するＤＦＧ化処理ステップと、データフローグラフの各ノードを、リコンフィギュラブル回路の論理回路に対応付けるマッピング処理ステップとを備えてもよい。

【００１２】

本発明のさらに別の態様は、複数の回路素子の機能および配置を決定して、所期の回路を生成する方法であって、回路素子の機能をノードとして表現して、入力データから出力データに至るデータの流れをノードの接続で表現するデータフローグラフを生成するＤＦＧ化処理ステップと、データフローグラフの各ノードを、生成するべき回路の回路素子の配置位置に対応付けるマッピング処理ステップとを備える回路生成方法を提供する。また本発明のさらに別の態様は、回路素子の機能をノードとして表現して入力データから出力データに至るデータの流れをノードの接続で表現するデータフローグラフを生成し、データフローグラフの各ノードを、生成するべき回路の回路素子の配置位置に対応付けて、その配置位置にノードの示す回路素子を形成することで生成された回路を提供する。

【００１３】
なお、以上の構成要素の任意の組合せ、本発明の表現を方法、装置、システム、記録媒体、コンピュータプログラムなどの間で変換したものもまた、本発明の態様として有効である。
【発明の効果】
【００１４】
本発明によれば、回路規模を削減したリコンフィギュラブル回路およびその周辺技術を提供することができる。
【発明を実施するための最良の形態】
【００１５】
（実施例１）
図１は、ＡＬＵアレイを用いたリコンフィギュラブルプロセッサの構成図を示す。図１に示すように、最初に構成情報格納メモリからＡＬＵの機能を制御する命令セットとＡＬＵ間の接続先を制御する接続データセット（以下、「構成情報」と呼ぶ）をＡＬＵアレイに設定する。ＡＬＵアレイは複数段で構成され、ＡＬＵを各段に複数個配置する。ＡＬＵには、予め複数の演算回路が実装されており、命令セットによってどの演算を行うかが選択される。上段から下段へのＡＬＵ間のデータ渡しは、ＡＬＵ間の接続切替を行う接続スイッチに接続データセットを設定することで、下段のどのＡＬＵにデータを渡すかが定められる。動作時には、構成情報に従って演算処理し、結果を出力する。
【００１６】
構成情報は、一般にＣ言語のような高品位言語で記述されたプログラムから作成される。Ｃのプログラムは、変換ツールによってＤＦＧ（Data Flow Graph）と呼ばれるデータフローグラフへ変換される。図２は、Ｃ言語プログラムの一例を示し、図３は、図２に示すプログラムに対応するＤＦＧを示す。図３において、mulは乗算、addは加算を示す。プログラムでは、変数ａ、ｂに入力された値をそれぞれ２倍、５倍して、その乗算結果を足し合わせる機能を表している。これをＤＦＧ化するには、ｘ、ｙを求める２つの式は並列に処理することが可能であるため、１段目でこの２つの乗算を行い、２段目でそれぞれの乗算結果を入力として加算を行う。変換ツールは、このＤＦＧからＡＬＵアレイ上の各ＡＬＵの命令とＡＬＵ間の接続を決定し、それらの情報からハードウェアの入力となるデータセットである構成情報に変換する。図４は、図３のＤＦＧから構成情報をＡＬＵアレイに割り当てた時の構成図を示す。ここで、movはＡＬＵへの入力をそのままスルーして、下段に渡すことを示す。
【００１７】
図５は、縦方向の段をまたぐＡＬＵ間接続をなくした接続方式を示す。図５に示す接続方式によると、段をまたいだ接続が存在しないため、回路規模を削減させることが可能となる。以下では、図５に示す接続方式よりも、さらに配線数を削減したＡＬＵアレイのアーキテクチャについて説明する。
【００１８】
（実施例２）
図６は、実施例２に係る処理装置１０の構成図である。処理装置１０は、集積回路装置２６を備える。集積回路装置２６は、回路構成を再構成可能とする機能を有する。集積回路装置２６は１チップとして構成され、リコンフィギュラブル回路１２、設定部１４、制御部１８、内部状態保持回路２０、出力回路２２および経路部２４を備える。リコンフィギュラブル回路１２は、設定を変更することにより、機能の変更を可能とする。
【００１９】
設定部１４は、第１設定部１４ａ、第２設定部１４ｂ、第３設定部１４ｃ、第４設定部１４ｄおよび選択器１６を有し、リコンフィギュラブル回路１２に所期の回路を構成するための構成情報４０を供給する。経路部２４は、フィードバックパスとして機能し、リコンフィギュラブル回路１２の出力を、リコンフィギュラブル回路１２の入力に接続する。内部状態保持回路２０および出力回路２２は、例えばデータフリップフロップ（Ｄ－ＦＦ

）などの順序回路あるいはメモリで構成され、リコンフィギュラブル回路１２の出力を受ける。内部状態保持回路２０は経路部２４に接続されている。リコンフィギュラブル回路１２は組合せ回路、またはＤ－ＦＦのような状態保持を含む順序回路として構成される。

【００２０】

リコンフィギュラブル回路１２は、機能の変更が可能なＡＬＵなどの論理回路の集合体を複数備えた構造を有し、また、それぞれの集合体の間に設けられて、集合体間の論理回路の接続を選択的に確立可能な少なくとも１つの接続部を有する。具体的に、リコンフィギュラブル回路１２において、演算機能を選択的に実行可能な複数のＡＬＵがマトリックス状に配列され、各段に配列された複数のＡＬＵが集合体を構成して、前段の集合体における処理結果が、接続部において選択的に確立される接続にしたがって後段の集合体に引き渡される。ここで、接続部は、接続する一方の集合体における論理回路が、別の集合体に含まれる一部の論理回路のみと接続可能であるように構成される。これにより、図５に示す接続方式と比較すると、回路規模を大幅に削減することが可能となる。各論理回路の機能と、論理回路間の接続関係は、設定部１４により供給される構成情報４０に基づいて設定される。構成情報４０は、以下の手順で生成される。

【００２１】

集積回路装置２６により実現されるべきプログラム３６が、記憶部３４に保持されている。プログラム３６は、信号処理回路または信号処理アルゴリズムなどをＣ言語などの高級言語で記述したものである。ＤＦＧ生成部３０は、ＤＦＧ化処理を実行し、具体的には記憶部３４に格納されたプログラム３６をコンパイルし、データフローグラフ（ＤＦＧ）３８に変換して記憶部３４に格納する。データフローグラフ３８は、入力変数および定数による入力データから出力データに至る演算ないしはデータの流れをグラフ構造で表現したものである。ここで、ＤＦＧ生成部３０は、リコンフィギュラブル回路１２における論理回路の集合体の接続制限に応じて、データフローグラフ３８を生成する。詳細については後述する。

【００２２】

構成情報生成部３２は、データフローグラフ３８から構成情報４０を生成する。構成情報４０は、データフローグラフ３８をリコンフィギュラブル回路１２にマッピングするためのデータであり、リコンフィギュラブル回路１２における論理回路の機能や論理回路間の接続関係を定める。本実施例では、ＤＦＧ生成部３０が、１つの回路を分割してできる複数の回路のサブＤＦＧを生成する機能をもつ。なお、この処理機能は、構成情報生成部３２により実現されてもよい。この場合は、構成情報生成部３２が、ＤＦＧ生成部３０が最初に生成したＤＦＧを分割して、複数の回路の構成情報を生成することになる。

【００２３】

図７は、１つの生成するべき回路４２を分割してできる複数の回路のサブＤＦＧ３８について説明するための図である。１つの回路４２を分割して生成される回路を、「分割回路」と呼ぶ。この例では、１つの回路４２が、４つの分割回路、すなわち分割回路Ａ、分割回路Ｂ、分割回路Ｃ、分割回路Ｄに分割されている。回路４２は、プログラム３６から直接生成されるデータフローグラフ３８における演算の流れにしたがって分割される。最初のデータフローグラフ３８において、上から下に向かう方向に演算の流れが表現される場合には、そのデータフローグラフ３８を上から所定の間隔で切り取り、その切り取った部分をサブＤＦＧとして設定する。流れにしたがって切り取る間隔は、リコンフィギュラブル回路１２における論理回路の段数以下に定められる。回路４２は、データフローグラフ３８の横方向で分割されてもよい。横方向に分割する幅は、リコンフィギュラブル回路１２における論理回路の１段あたりの個数以下に定められる。

【００２４】

特に、生成すべき回路がリコンフィギュラブル回路１２よりも大きい場合に、ＤＦＧ生成部３０は、リコンフィギュラブル回路１２にマッピングできる大きさになるように、回路４２を分割することが好ましい。なお、この処理は、構成情報生成部３２によって、構成情報を再生成するという形で行われてもよい。ＤＦＧ生成部３０は、リコンフィギュラ

ブル回路１２における論理回路の配列構造と、プログラム３６から直接生成されるデータフローグラフ３８によって、回路４２の分割方法を定める。リコンフィギュラブル回路１２の配列構造は、制御部１８からＤＦＧ生成部３０に伝えられてもよく、また予め記憶部３４に記録されていてもよい。また、制御部１８が、回路４２の分割方法をＤＦＧ生成部３０に指示してもよい。このようにして生成されたサブＤＦＧは、構成情報生成部３２により構成情報に変換される。

【００２５】

以上の手順を実行することにより、記憶部３４は、リコンフィギュラブル回路１２を所期の回路として構成するための複数の構成情報４０を記憶する。複数の構成情報４０は、分割回路Ａを構成するための構成情報４０ａ、分割回路Ｂを構成するための構成情報４０ｂ、分割回路Ｃを構成するための構成情報４０ｃ、および分割回路Ｄを構成するための構成情報４０ｄである。複数の構成情報４０は、１つの回路４２を分割した複数の分割回路をそれぞれ表現したものである。このように、リコンフィギュラブル回路１２の回路規模に応じて、生成すべき回路４２のサブＤＦＧ３８ないしは構成情報４０を生成することにより、汎用性の高い処理装置１０を実現することが可能となる。別の視点からみると、本実施例の処理装置１０によれば、回路規模の小さいリコンフィギュラブル回路１２を用いて、所望の回路を再構成することが可能となる。

【００２６】

図８は、実施例２による接続方式のリコンフィギュラブル回路１２の構成図である。リコンフィギュラブル回路１２は、複数の論理回路５０が複数段にわたって配列されたもので、各段に設けられた接続部５２によって、前段の論理回路列の出力と後段の論理回路列の入力が設定により接続可能な構造となっている。各段に設けられた複数の論理回路５０は集合体を構成する。ここでは、論理回路５０の例としてＡＬＵを示す。各ＡＬＵは、論理和、論理積、ビットシフトなどの複数種類の多ビット演算を設定により選択的に実行できる。各ＡＬＵは、複数の演算機能を選択するためのセレクタを有している。

【００２７】

図８に示すリコンフィギュラブル回路１２では、横方向に６個、縦方向に３個のＡＬＵが配置されたＡＬＵアレイとして構成される。第１段のＡＬＵ００、ＡＬＵ１０、・・・、ＡＬＵ５０には、入力変数や定数が入力され、設定された所定の演算がなされる。演算結果の出力は、第１段の接続部５２に設定された接続にしたがって、第２段のＡＬＵ０１、ＡＬＵ１１、・・・、ＡＬＵ５１に入力される。第１段の接続部５２においては、第１段のＡＬＵ列の出力と第２段のＡＬＵ列の入力の間で、一定の接続制限が課された接続関係を実現できるように結線が構成されており、設定により、その範囲内での所期の結線が有効となる。なお、第２段の接続部５２においても同様である。最終段である第３段のＡＬＵ列は演算の最終結果を出力する。接続部５２は、ＡＬＵ段の間で、物理的に近接して配置された論理回路同士を接続可能とするように構成される。これにより、配線長を短くすることができ、回路規模を削減することができる。その結果、低消費電力化及び処理高速化が可能となる。

【００２８】

図８に示すリコンフィギュラブル回路１２では、３段×６列のＡＬＵが存在し、上段における１つのＡＬＵからの配線は、下段の３つのＡＬＵに制限される。図示のように、下段における１つのＡＬＵの入力は、上段における直上のＡＬＵと、直上のＡＬＵの左右のＡＬＵに制限され、また上段における１つのＡＬＵの出力は、下段における直下のＡＬＵと、直下のＡＬＵの左右のＡＬＵに制限される。例えば、ＡＬＵ２１に関してみると、その入力は、ＡＬＵ１０、ＡＬＵ２０、ＡＬＵ３０の３方向に制限され、その出力は、ＡＬＵ１２、ＡＬＵ２２、ＡＬＵ３２の３方向に制限される。なお、左または右に対応するＡＬＵが存在しなければ、その入力および出力は、それぞれ２方向に制限される。このような配線とすることにより、１段につき６個のＡＬＵで３段構成とした場合、ＡＬＵ間の配線数は、図５に示した接続方式の配線数と比較すると、

（図５に示すＡＬＵアレイの配線数）

６×１２＝７２
（図８）に示すＡＬＵアレイの配線数）
３×８＋２×４＝３２
となり、およそ５０％の配線数の削減が可能となる。
【００２９】
図９は、図２に示すＣプログラムを図８に示したＡＬＵアレイに割り当てた（マッピングした）結果を示す。図９では、可能な限り上方のＡＬＵを使用している。図９を参照して、□はＡＬＵを示し、また□を塗り潰した■は命令(この例ではmul、add)を割り当てたＡＬＵを示し、太線で示された線は、ＡＬＵ間の接続を示す。以下、このように、有意な命令を割り当てられるＡＬＵをノードとして表現する。なお、本実施例および以降の実施例において、ノードは、ＡＬＵの機能を特定するための探索処理において演算的に使用する概念である。また、同様に、マッピング（処理）は、リコンフィギュラブル回路１２上に回路を実際に構成する処理だけでなく、演算上、ＡＬＵの機能を特定するための探索処理において演算的に実行する処理も含む。演算上のマッピング処理は、制御部１８により制御されてもよく、またＤＦＧ生成部３０ないしは構成情報生成部３２により制御されてもよい。
【００３０】
図１０は、データフローグラフ３８の例を示す図である。データフローグラフ３８においては、入力される変数や定数の演算の流れが段階的にグラフ構造で表現されている。図中、演算子は丸印で示されている。構成情報生成部３２は、このデータフローグラフ３８をリコンフィギュラブル回路１２にマッピングするための構成情報４０を生成する。実施例では、特にデータフローグラフ３８をリコンフィギュラブル回路１２にマッピングしきれない場合に、データフローグラフ３８を複数の領域に分割して、分割回路のサブＤＦＧ３８ないしは構成情報４０を生成する。データフローグラフ３８による演算の流れをリコンフィギュラブル回路１２上で実現するべく、構成情報４０は、演算機能を割り当てる論理回路を特定し、また論理回路間の接続関係を定め、さらに入力変数や入力定数などを定義したデータとなる。したがって、構成情報４０は、各論理回路５０の機能を選択するセレクタに供給する選択情報、接続部５２の結線を設定する接続情報、必要な変数データや定数データなどを含んで構成される。
【００３１】
図６に戻って、回路の構成時、制御部１８は、１つの回路を構成するための構成情報４０を選択する。ここでは、制御部１８が、図７に示す回路４２を構成するための構成情報４０、すなわち分割回路Ａの構成情報４０ａ、分割回路Ｂの構成情報４０ｂ、分割回路Ｃの構成情報４０ｃおよび分割回路Ｄの構成情報４０ｄを選択するものとする。制御部１８は、選択した構成情報４０を設定部１４に供給する。設定部１４はキャッシュメモリや他の種類のメモリを有し、供給される構成情報４０をそれぞれ保持する。具体的に制御部１８は、構成情報４０ａを第１設定部１４ａに、構成情報４０ｂを第２設定部１４ｂに、構成情報４０ｃを第３設定部１４ｃに、構成情報４０ｄを第４設定部１４ｄに供給する。
【００３２】
設定部１４は、選択された構成情報４０をリコンフィギュラブル回路１２に設定し、リコンフィギュラブル回路１２の回路を再構成する。これにより、リコンフィギュラブル回路１２は、所期の演算を実行できる。リコンフィギュラブル回路１２は、基本セルとして高性能の演算能力のあるＡＬＵを用いており、またリコンフィギュラブル回路１２および設定部１４を１チップ上に構成することから、コンフィグレーションを高速に、例えば１クロックで実現することができる。制御部１８はクロック機能を有し、クロック信号は、内部状態保持回路２０および出力回路２２に供給される。また制御部１８はカウンタ回路を含み、カウント信号を選択器１６に供給してもよい。この場合、カウンタ回路は４進カウンタである。
【００３３】
図１１は、実施例における信号処理のフローチャートを示す。制御部１８は、カウンタ

回路からのカウント信号に合わせて、リコンフィギュラブル回路１２に複数の構成情報４０、すなわち構成情報４０ａ、構成情報４０ｂ、構成情報４０ｃおよび構成情報４０ｄを順次供給するように設定部１４を制御する。設定部１４が、複数の構成情報４０をリコンフィギュラブル回路１２に順次供給することにより、全体として１つの回路が構成されることになる。出力回路２２は、設定部１４によりリコンフィギュラブル回路１２が複数回、ここでは４回構成されると、リコンフィギュラブル回路１２の出力を出力する。この回数は、使用する構成情報４０のサイクルに相当する。以下、具体的な手順を示す。

【００３４】

まず、制御部１８が、選択器１６を制御して第１設定部１４ａを選択する。選択器１６は、カウンタ回路により制御されてもよい。第１設定部１４ａは、分割回路Ａの構成情報４０ａをリコンフィギュラブル回路１２に供給し、リコンフィギュラブル回路１２上に分割回路Ａを構成する（Ｓ１０）。分割回路Ａが構成されると同時に、入力データが分割回路Ａに供給される。組合せ回路である分割回路Ａは、次のクロック信号までの間に、演算処理を実行する。

【００３５】

制御部１８がクロック信号を内部状態保持回路２０に供給すると、内部状態保持回路２０は、分割回路Ａによる処理結果を保持する（Ｓ１２）。Ｓ１０およびＳ１２のステップを第１サイクルと呼ぶ。同時に、制御部１８が、選択器１６を制御して第２設定部１４ｂを選択する。第２設定部１４ｂは、分割回路Ｂの構成情報４０ｂをリコンフィギュラブル回路１２に供給し、リコンフィギュラブル回路１２上に分割回路Ｂを構成する。このとき、内部状態保持回路２０に保持された分割回路Ａの処理結果が、経路部２４を通って分割回路Ｂの入力に供給される（Ｓ１４）。分割回路Ｂは、次のクロック信号までの間に、演算処理を実行する。

【００３６】

制御部１８が次のクロック信号を内部状態保持回路２０に供給すると、内部状態保持回路２０は、分割回路Ｂの処理結果を保持する（Ｓ１６）。Ｓ１４およびＳ１６のステップを第２サイクルと呼ぶ。同時に、制御部１８が、選択器１６を制御して第３設定部１４ｃを選択する。第３設定部１４ｃは、分割回路Ｃの構成情報４０ｃをリコンフィギュラブル回路１２に供給し、リコンフィギュラブル回路１２上に分割回路Ｃを構成する。このとき、内部状態保持回路２０に保持された分割回路Ｂの処理結果が、経路部２４を通って分割回路Ｃの入力に供給される（Ｓ１８）。分割回路Ｃは、次のクロック信号までの間に、演算処理を実行する。

【００３７】

制御部１８が次のクロック信号を内部状態保持回路２０に供給すると、内部状態保持回路２０は、分割回路Ｃの処理結果を保持する（Ｓ２０）。Ｓ１８およびＳ２０のステップを第３サイクルと呼ぶ。同時に、制御部１８が、選択器１６を制御して第４設定部１４ｄを選択する。第４設定部１４ｄは、分割回路Ｄの構成情報４０ｄをリコンフィギュラブル回路１２に供給し、リコンフィギュラブル回路１２上に分割回路Ｄを構成する。このとき、内部状態保持回路２０に保持された分割回路Ｃの処理結果が、経路部２４を通って分割回路Ｄの入力に供給される（Ｓ２２）。分割回路Ｄは、次のクロック信号までの間に、演算処理を実行する。

【００３８】

制御部１８が次のクロック信号を出力回路２２に供給すると、出力回路２２は、分割回路Ｄの処理結果を出力する（Ｓ２４）。Ｓ２２およびＳ２４のステップを第４サイクルと呼ぶ。第１サイクルから第４サイクルまでの処理を繰り返し行う場合には、再度、制御部１８が選択器１６を制御して第１設定部１４ａを選択し、リコンフィギュラブル回路１２上に分割回路Ａを構成して、入力データが供給される。

【００３９】

以上のように、１つの回路４２を分割した複数の分割回路Ａ～Ｄをリコンフィギュラブル回路１２上に順次構成し、各分割回路の出力を次の分割回路の入力にフィードバックし

て各分割回路における演算処理を実行し、最後に構成された分割回路Ｄから、回路４２の出力を取り出す。Ｓ１０からＳ２４までにかかる時間は４クロック分であり、本実施例の処理装置１０によると、限られたリコンフィギュラブル回路１２の回路規模のなかで、効率よい演算処理を実行することができる。また、リコンフィギュラブル回路１２の回路規模が小さいため、消費電力も小さくできる。

【００４０】

制御部１８は、内部状態保持回路２０および出力回路２２に同一のクロック信号を供給してもよいが、出力回路２２に供給するクロック信号の周期を、内部状態保持回路２０に供給するクロック信号の周期の４倍に設定してもよい。内部状態保持回路２０および出力回路２２に同一のクロック信号を供給する場合は、内部状態保持回路２０に出力回路２２の役目をもたせ、１つの回路にまとめることもできる。この場合は、出力先の回路以降で必要な信号を取り出すための回路が必要となる。また、この例では第１設定部１４ａ～第４設定部１４ｄの４つの設定部を利用したが、この数も回路４２の分割数に応じて変動することは当業者に容易に理解されるところである。以下の実施例は、基本的に、上記した実施例２に関連して説明した処理装置１０の構造をベースとする。

【００４１】

（実施例３）

図２のＣプログラムから作成されたＤＦＧは、リコンフィギュラブル回路１２におけるＡＬＵアレイのサイズ（この場合、横６個×縦３個）に収まっている。実施例３では、ＤＦＧがこのＡＬＵアレイサイズを超える場合の処理について述べる。ＤＦＧがＡＬＵアレイの縦方向のサイズを超える場合、ＤＦＧをＡＬＵアレイの縦方向に分割する。

【００４２】

図１２は、ＡＬＵアレイの縦方向のサイズを超えたＤＦＧをＡＬＵアレイにマッピングした仮想的な状態の一例を示す。このＤＦＧは縦の大きさが５であるので、ＤＦＧ生成部３０は、このＤＦＧを２つに分割してできる分割回路のサブＤＦＧを生成する。

【００４３】

図１３は、図１２におけるＤＦＧを２つに分割したサブＤＦＧをＡＬＵアレイにマッピングした状態を示す。図１３（ａ）は、図１２に示した前半３段分のＡＬＵアレイのマッピング状態であり、図１３（ｂ）は、図１２に示した後半２段分のＡＬＵアレイのマッピング状態である。ここでは、分割したＤＦＧを、サブＤＦＧと呼ぶ。サブＤＦＧをＡＬＵアレイに割り当てて動作させる場合、図１３（ａ）のＡＬＵアレイの３段目の出力は、一旦、内部状態保持回路２０（図６参照）に保持されて、経路部２４のループ配線を通じて図１３（ｂ）のＡＬＵアレイの１段目の入力データとして引き渡される。具体的には、制御部１８が、設定部１４に対して、第１設定部１４ａに、図１３（ａ）に示すＡＬＵアレイを構成するための構成情報を保持させ、また第２設定部１４ｂに、図１３（ｂ）に示すＡＬＵアレイを構成するための構成情報を保持させて、選択器１６を介して、リコンフィギュラブル回路１２（ＡＬＵアレイ）に、順次、構成情報を用いて分割回路を構成させることで、図１２に示したＡＬＵアレイの処理を実現することができる。その結果、回路規模の小型化が実現でき、低消費電力化が可能となる。また、リコンフィギュラブル回路１２におけるＡＬＵアレイのサイズを超えるＤＦＧを処理する場合でも、ＡＬＵアレイを設計し直す必要がなく、回路の再利用性が高まる。

【００４４】

（実施例４）

次に、ＤＦＧがＡＬＵアレイの横方向のサイズを超える場合、ＤＦＧを横方向に分割する。このＤＦＧの分割は、ＤＦＧ生成部３０により行われる。

図１４は、縦横の両方向にＡＬＵアレイのサイズを超えるＤＦＧをＡＬＵアレイにマッピングした仮想的な状態の一例を示す。このＤＦＧは、横８個×縦５個のサイズで表現される。ＡＬＵアレイのサイズは、横６個×縦３個であるため、このＤＦＧを、３つのサブＤＦＧに分割する。図１４において、点線で３つに分割されたサブＤＦＧａ、サブＤＦＧｂおよびサブＤＦＧｃで、もとのＤＦＧを構成させる。

【００４５】
図１５は、図１４におけるＤＦＧを分割したそれぞれのサブＤＦＧをＡＬＵアレイにマッピングした状態を示す。図１５（ａ）は、サブＤＦＧａをマッピングした状態を示し、図１５（ｂ）は、サブＤＦＧｂをマッピングした状態を示し、図１５（ｃ）は、サブＤＦＧｃをマッピングした状態を示す。サブＤＦＧをＡＬＵアレイに割り当てて動作させる場合、図１５（ｃ）のＡＬＵアレイの処理は、図１５（ａ）および図１５（ｂ）に示すＡＬＵアレイの出力を入力データとしているため、設定部１４において、構成情報をＡＬＵアレイに割り当てる順序は、サブＤＦＧａの構成情報、サブＤＦＧｂの構成情報、サブＤＦＧｃの構成情報の順に行う必要がある。なお、サブＤＦＧａとサブＤＦＧｂの構成情報の割り当て順序は入れ替わってもよい。また、サブＤＦＧａおよびサブＤＦＧｂで分割回路を構成したときの結果は、内部状態保持回路２０に蓄えられ、サブＤＦＧｃの割り当て時に、入力データとして読み込まれる。このように、縦横のサイズがＡＬＵアレイのサイズを超えるＤＦＧを処理する場合においても、小型の回路で実現でき、低消費電力化を可能とする。また、リコンフィギュラブル回路１２におけるＡＬＵアレイのサイズを超えるＤＦＧを処理する場合でも、ＡＬＵアレイを設計し直す必要がなく、回路の再利用性が高まる。

【００４６】
（実施例５）
上述した実施例３および実施例４では、ＤＦＧをそのまま分割するだけで、サブＤＦＧとしてＡＬＵアレイに割り当てられる場合について述べた。実施例５では、ＤＦＧがＡＬＵアレイの接続制限に合わない場合の処理方法について述べる。この処理は、ＤＦＧ生成部３０により行われる。なお、この処理は、制御部１８が担当してもよい。

【００４７】
図８に関連して説明したように、ＡＬＵアレイの接続制限が３方向の場合、ノードが４つ以上の入力や出力を持つと、そのノードをＡＬＵアレイに直接割り当てできない。入力については、とり得る命令の処理内容が問題となる。表１は、命令セットの例を示す。ここで、merge命令は、条件文からの出力(0か1)によって、２つのデータのうちの１つを選択する処理を行うため、３入力が必要となる。ここに示した命令であれば、全て３入力以下となるため、入力用の接続は全て制限内に収まる。

【表１】

表1

命令	入力データ数	処理内容
mov	1	スルー
add	2	加算
sub	2	減算
mul	2	乗算
div	2	除算
mod	2	剰余
not	1	否定
and	2	&（ビット毎）
or	2	|（ビット毎）
xor	2	⁻（ビット毎）
neg	1	符号逆転
asr	2	右シフト
lsl	2	左シフト
beq	2	==（条件文）
bne	2	!=（条件文）
bgt	2	＞（条件文）
bge	2	>=（条件文）
merge	3	条件マージ

出力については、もとのＣのプログラムによって変化するため、実施例のリコンフィギュラブル回路１２においては、ノードの出力数が３以下となるように、出力数を調整する必要がある。

【００４８】

図１６は、Ｃプログラムの例を示し、図１７は、図１６に示すＣプログラムのＤＦＧを示す。図１７のnegノードの出力数は４であるため、そのままではＤＦＧをＡＬＵアレイに割り当てることはできない。この場合の処理方法として、以下に３つの方法を示す。

【００４９】

（ｉ）ノードコピー

１番目の方法は、接続制限に合わないノードをコピーし、出力数を分散させる方法であ

る。ここでは、ＤＦＧにおいて、入力を同一とする別のノードを複製して、ノード出力を分散させる処理を実行する。図１６の例では、negノードを同じ段にコピーすることにより、１ＡＬＵあたりの出力数を３と１に制限する。

【００５０】

図１８は、ノードコピーを行って、出力数を３つに制限したＤＦＧを示す。１段目のnegノードを、出力数が３つと１つになるように複製している。これにより、リコンフィギュラブル回路１２における接続制限の問題を解決することができる。また、後述する（ii）スルーノードの挿入、および（iii）ＤＦＧ分割の手法と比較すると、これらの手法に必要な段数の増加をなくすことができ、処理の高速化が可能となる。

図１９は、図１８に示したＤＦＧを割り当てたＡＬＵアレイを示す。なお、ここでは、説明の便宜上、横８×縦５のＡＬＵアレイとして示しているが、前述したように、このＡＬＵアレイは、横６×縦３のＡＬＵアレイに分割される。

【００５１】

（ii）スルーノードの挿入

２番目の方法は、ノード間にスルーノードを挿入することによって、隣接するＡＬＵ間の配線に適したＤＦＧに変形する方法である。ここでは、ＤＦＧにおいて、スルーノードを、ノードの出力ノードの一部とともに、ノードの出力と接続するように追加し、スルーノードの後に残りの出力ノードを配置することで、ノード出力を分散させる処理を実行する。スルーノードは表１のmov命令を使用し、このノードは、何の処理も行わない。

【００５２】

図２０は、movノードを挿入したＤＦＧを示す。これは、図１７における２段目のnegノードの出力を用いた処理を、スルーノードを用いて一部を３段目にずらすことで、４つの処理を２段目と３段目で分けて行うＤＦＧとして構成されている。これにより、リコンフィギュラブル回路１２における接続制限の問題を解決することができる。また、後述の（iii）ＤＦＧ分割の手法の場合に比べて、１つのＤＦＧとして処理が可能であるため、処理の高速化が可能となる。

【００５３】

図２１は、図２０に示したＤＦＧを割り当てたＡＬＵアレイを示す。なお、ここでは、説明の便宜上、横８×縦５のＡＬＵアレイとして示しているが、前述したように、このＡＬＵアレイは、横６×縦３のＡＬＵアレイに分割される。

【００５４】

（iii）ＤＦＧ分割

３番目の方法は、接続制限に合わないノードに接続されたノードを出力制限数だけ配置し、サブＤＦＧとして切り出す方法である。残りのノードは、別のサブＤＦＧとして作成する。

【００５５】

図１７の例では、negノードに接続された２つのaddと１つのsubを同じサブＤＦＧとして作成する。残り１つのsubとその出力であるaddノード以降は割り当てできないので、別のサブＤＦＧとして作成する。

図２２は、分割した２つのサブＤＦＧを示す。図２３は、図２２に示したサブＤＦＧを割り当てたＡＬＵアレイを示す。ＤＦＧの分割については、後に詳述する。

【００５６】

（実施例６）

次に、ＤＦＧのノードの接続形態によってＤＦＧがＡＬＵアレイに割り当てできない場合の処理方法について述べる。

図２４は、条件文を含んだＣプログラムの一例を示し、図２５は、図２４のＣプログラムに対応したＤＦＧを示す。条件文に対するＤＦＧでは、条件ノードに対応してmergeノードが生成され、条件ノードの出力の真偽に応じて、mergeノードの２つの入力のうちの１つが選択される。従って、mergeノードの入力数は３である。

【００５７】

図２６は、図２５のＤＦＧをそのまま割り当てたＡＬＵアレイを示す。図２６に示すように、２つのmergeノードは、それぞれＡＬＵ１１およびＡＬＵ４１（図８参照）に対応付けられるが、その下段においてＡＬＵ１１およびＡＬＵ４１の出力先は、接続制限のため合わせることができない。したがって、図２５のＤＦＧにおいて、addノードをＡＬＵアレイに対応付けることができない。この場合、上述したＤＦＧ分割の方法を用いる。

【００５８】

図２５の左側のmergeノードまでの領域と残りのノードの領域とを２つのサブＤＦＧとして分割する。図２７は、このようにして生成したサブＤＦＧを割り当てたＡＬＵアレイを示す。このように、ＤＦＧがＡＬＵアレイの接続制限に合わない場合でも、ＤＦＧを分割することにより、接続制限したＡＬＵアレイを用いた処理が可能となる。

【００５９】

（実施例７）

前述したように、ＤＦＧをＡＬＵアレイに割り当てるには、ＡＬＵアレイの大きさや接続制限を考慮して割り当てなければならない。実施例７では、ＤＦＧ内の各ノードをどの位置のＡＬＵに割り当てるか決定する方法、すなわち、接続制限されたリコンフィギュラブル回路１２における論理回路の機能決定方法について述べる。ここで、各命令間の入出力変数を元に、ノード間の接続関係が明らかになったＤＦＧを初期状態のＤＦＧと呼ぶ。この状態からＡＬＵアレイへ割り当てるには、以下の手順で行う。

（１）ノード位置の高さ決定
（２）フライパス除去
（３）４以上の出力数を持つノードの最適化
（４）ノードの横方向位置決定、ＤＦＧ分割
（５）サブＤＦＧの合成

なお、ＤＦＧにおける各ノードをリコンフィギュラブル回路１２の論理回路に対応付けるマッピング処理は、制御部１８により行われてもよく、またＤＦＧ生成部３０により行われてもよい。なお、このマッピング処理は、実際にリコンフィギュラブル回路１２上に回路を構成するものではなく、回路を構成するための最終的なサブＤＦＧや構成情報を取得するために、演算上実行されるマッピング処理を意味する。

【００６０】

（１）では、既に公知のアルゴリズムを適用することにより、ノード位置の高さを決める。

（２）では、ノード間の接続が段を跨ぐ場合、スルーノードを挿入する。

（３）では、ノード出力数が４以上の場合、ノードをコピーし、３以下に削減する。

（４）では、ノードを横方向に探索することでノードの横方向位置を決定し、配置できないＤＦＧは分割を行う。この分割後のＤＦＧをサブＤＦＧと呼ぶ。

（５）では、複数のサブＤＦＧのうち、並列的に動作可能で、かつ、ＡＬＵアレイにまとめて配置可能なものを合成する。

【００６１】

（４）の横方向の探索を行う場合、各段に対して全探索を行うと、膨大な処理時間を要する。例えば、１段あたり６個のＡＬＵが配置されるＡＬＵアレイに、１段あたり３個のノードで３段のＤＦＧの各ノード横方向位置の探索を行うとすると、

$({}_6P_3)^3 = (6\times5\times4)^3 = 120^3 = 1{,}728{,}000$(回)

の位置探索を行う必要がある。また、この探索では全パターンを探索しても必ず全ノードが配置できるとは限らず、サブＤＦＧへの分割も必要である。このため、効率的な探索方法が必要となる。

【００６２】

図２８は、図１８に示すＤＦＧの各ノードにノード番号を割り振った状態を示す。ノード番号の割り振り方法は任意であってよいが、この例では、基本的に左から右に番号を増やすように割り振っている。まず、最上段のノードから１つを選択し、ＡＬＵアレイ内の左上に配置する。複数のノードがある場合は、どれを選んでも構わないが、ここでは、ノ

ード番号の小さいものを選択した場合を例示する（具体的にはノード１を選択する）。図２９は、左上にノード１を配置した状態を示す。なお、以下では、説明の便宜上、図８に示したリコンフィギュラブル回路１２のＡＬＵアレイよりも段数の多いＡＬＵアレイを例に、説明を行う。既述したように、実際に図８のＡＬＵアレイに割り当てる場合には、回路を分割する必要がある。

【００６３】

次に、配置したノードの出力を入力に持つノードから１つを選択し、配置したノードの左下、真下、右下の順に配置可能ならば配置する。ここでは、ノード１の出力を入力に持つノードが、ノード２，３，４であるが、ここでもノード番号の小さいものを選択した場合を例示する。今後、位置を(X,Y)で表すことにする。ノード１の位置は(0,0)であるので、ノード２は(-1,1)、(0,1)、(1,1)の順に配置可能な位置を探索するが、負値はとることができないため、(0,1)に配置する。次に、ノード２の入力に、ノード１以外が存在するか否かを調べて、存在していれば、入力ノードを配置する。ここでは、ノード１以外に存在しないため、このステップは省略する。次に、ノード２の出力を入力に持つノードはノード６であるので、これも同様にして、(0,2)に配置する。ノード６はその入力に、未配置のノード３があるため、次にこの配置を行う。ノード３が配置できる位置を、ノード６の左上、真上、右上の順に探索する。ここで、既に真上の(0,1)にはノード２が配置済みのため、ノード３は(1,1)に配置する。更に、ノード３の入力にはノード１がある。ノード１はノード３の左上になるので、マージ可能である。ここまで配置した状態を図３０に示す。

【００６４】

同様にして、ノード６の出力を入力に持つノード８を配置する。ノード８を真下、右下、と順に移動し、ノード８が全て配置可能な位置を探索する。ノード８を(0,3)に置いた図を図３１に、(1,3)に置いた図を図３２に示す。図３１および図３２に示すように、ノード７を(1,2)又は(2,2)のどちらに置いてもその入力のノード４は(2,1)に置くことになり、ノード４の入力であるノード１とはマージできない。よって、この時点でノード８の配置は不適切であったことが判明し、ノード８、ノード７及びノード４の横位置を削除する。

【００６５】

次に、図３３に示すように、ノード６を(1,2)に変更し、ノード８の配置探索を行う。なお、この場合であっても、上記した探索処理を繰り返すと、ノード４を(2,1)に配置するしかないため、ノード４の入力であるノード１とはマージできない。このような探索処理を順次繰り返す。ノード２を(0,1)から(1,1)に変更して同様の探索を行い、全て配置不可であれば、ノード１の配置ステップまで戻って、ノード１の位置を(0,0)から(1,0)、(2,0)、・・・、(5,0)まで変更して探索する。この例の場合、結果として、ノード１の位置を(1,0)にずらすと、全てのノードを配置できることになる。この処理アルゴリズムから明らかなように、各段に対して全探索を行う場合と比較すると、大幅に探索数を低減することが可能となる。

【００６６】

図３４は、ノード１を(1,0)に配置して、ノード２、ノード６、ノード３の順にそれぞれの配置を決定した状態を示す。続いて、図３５に示すように、ノード６に対して、ノード８、ノード７、ノード４の配置を決定する。しかしながら、この場合、ノード７に対してノード５を配置することができない。したがって、ノード８の位置を１つ右にずらして配置してみる。そうすると、図３６に示すように、ノード７を(2,2)に配置することができる。したがって、ノード５を(3,1)に配置することができ、図３７に示すように、接続制限されたＡＬＵアレイにおける配置を決定することができる。

【００６７】

以下、図３８から図４１を用いて、接続制限されたＡＬＵアレイにおけるノード配置方法を説明する。ノードは論理回路（ＡＬＵ）の機能を表現するものであり、ノードの配置を定めることは、すなわちＡＬＵアレイにおける論理回路の機能を定めることと等価であ

る。以下では、図２８に示したＤＦＧをＡＬＵアレイにマッピングする場合を例に説明する。

【００６８】

図３８は、ノード配置のメインフローを示す。図３８において、ＤＦＧにおける最上段のノードN_0を１つ取り出す（Ｓ１０）。ここでは、ノード１を取り出すものとする。X_0およびY_0の初期値をそれぞれ０に設定し（Ｓ１２）、ノードN_0を(0,0)に配置する（Ｓ１４）。この状態は、図２９に示される。

【００６９】

次に、ノードN_0の出力ノードN_1を１つ取り出す（Ｓ１６）。ただし、既に取り出したノードがある場合は、それを除く。図２８を参照すると、ノード１の出力ノードとしてノード２、ノード３、ノード４が存在するが、ここではノード２を取り出すものとする。出力ノードN_1が存在する場合（Ｓ１８のＮ）、処理は、出力ノード配置モードにはいる（Ｓ２０）。なお、（N_1，X_0，Y_0）は、出力ノード配置モードにおける処理フローに引数として渡される。

【００７０】

図３９は、出力ノード配置モードの処理フローを示す。出力ノード配置モードにおいて、まず、段数を１段下げて（Y'←Y+1）、列を１つだけ左にずらす（X'←X-1）（Ｓ３０）。その結果、座標(0,0)は、座標(-1,1)に変換される。段数を１段下げるのは、出力ノードが１段下に存在するためである。列を１つ左にずらすのは、本実施例のＡＬＵアレイは、出力ノードの存在方向を、直下とその左右の３方向に制限しているためであり、まず最初に左下にノードを配置できるかどうかを探索して、続いて直下、右下にノードを配置した場合を順をおって探索するためである。この状態で、入力ノード配置モードにはいる（Ｓ３２）、なお、（Ｎ，X'，Y'）は、入力ノード配置モードにおける処理フローに引数として渡される。

【００７１】

図４０は、入力ノード配置モードの処理フローを示す。入力ノード配置モードにおいて、X＜0またはX_{MAX}＜Xの場合（Ｓ５０のＹ）、配置失敗として入力ノード配置モードを終了し、図３９のＳ３４に移行する。なお、X_{MAX}はＡＬＵアレイ１段におけるＡＬＵの個数から１を減じた値であり、図８に示す例では、ＡＬＵの横方向の個数が６であるため、X_{MAX}は５である。また、0≦X≦X_{MAX}であって（Ｓ５０のＮ）、(X,Y)にノードが既に配置されている場合（Ｓ５２のＹ）、配置されているノードが自分自身であれば（Ｓ５４のＹ）、配置成功であり、自分自身でなければ（Ｓ５４のＮ）、配置失敗となる。Ｓ５４の判定が終了すると、図３９のＳ３４に移行する。図２８のＤＦＧをマッピングする具体例においては、まず、引数として(-1,1)が渡されるため、Ｓ５０においてX＜0であることが判定されるために、配置失敗として、図３９のＳ３４に戻る。

【００７２】

図３９のＳ３４において、配置失敗であったために（Ｓ３４のＮ）、X'を１インクリメントする（Ｓ３６）。この処理は、出力ノードを、左下から直下、右下へと順次サーチするためのステップであり、この例では、続いて、直下にノードを配置した場合の妥当性の検証を行うことになる。具体的に、１インクリメントされると、座標(X',Y')は(0,1)となる。X'が、(X+1)を超えない限りにおいて（Ｓ３８のＮ）、入力ノード配置モードに戻ることができる（Ｓ３２）。なおX'が、(X+1)を超える場合（Ｓ３８のＹ）とは、出力ノードとして、右下よりさらに右側の出力ノードを探索する場合に相当する。本実施例では、論理回路の接続に制限を課していることを前提としているため、直下から２つ以上右側（ないしは左側）への配置は考慮しない。

【００７３】

Ｓ３２において、再度、入力ノード配置モードにはいる。このとき、Ｓ５０では、X(=0)が0以上の値をとっており、且つX_{MAX}以下であり（Ｓ５０のＮ）、さらに(X,Y)、具体的には(0,1)にノードを配置済みでないため（Ｓ５２のＮ）、ノードＮ（ここでは、ノード２）を(X,Y)に配置する（Ｓ５６）。これにて、ノード１が(0,0)に、ノード２が(0,1)に

配置された状態となる。

【００７４】

続いて、ノードＮの入力ノードＮ’をＤＦＧから１つ取り出す（Ｓ５８）。なお、既に取り出した入力ノードは除く。ここで、ノード２の入力ノードはノード１のみであり（取り出し済み）、他の入力ノードは存在しないため、入力ノードＮ’がない（Ｓ６０のＹ）とし、ここまでの配置を成功とみなして、入力ノード配置モードを抜けて、図３９のＳ３４に移行する。Ｓ３４では、配置可能であったため（Ｓ３４のＹ）、ノード配置チェック・配置状態保存モード（以下、「ノード配置チェックモード」と呼ぶ）に移行する（Ｓ４０）。

【００７５】

図４１は、ノード配置チェックモードの処理フローを示す。このノード配置チェックモードでは、最もノード数を多く配置することのできた状態をチェックして、配置状態を保存する。全てのノードを配置することができた場合だけでなく、一部のみしか配置できない場合であっても、その配置状態を保存しておく。これは、例えば複数のサブＤＦＧのデータとしてもつことができる。これにより、後にＡＬＵアレイを構成する場合に、生成すべき回路を１つのＤＦＧによって作成することができない場合であっても、サブＤＦＧを組み合わせることで、生成すべき回路を作成することが可能となる。

【００７６】

ノード配置チェックモードにおいて、まず、現在配置しているノード数Ｓをカウントする（Ｓ８０）。ノード数Ｓは、S_{MAX}と比較される。S_{MAX}の初期値は0にセットされている。ノード数ＳがS_{MAX}よりも大きい場合（Ｓ８２のＹ）、そのときのノード配置を保持して（Ｓ８４）、S_{MAX}をＳに設定する（Ｓ８６）。なお、ノード数ＳがS_{MAX}以下の場合（Ｓ８２のＮ）、ノード配置チェックモードを終了して、図３９のＳ４２に移行する。

【００７７】

Ｓ４２において、ノードＮの出力ノードＮ’をＤＦＧから１つ取り出す。具体例では、ノード２の出力ノード６を取り出すこととする。出力ノードＮ’がない場合（Ｓ４４のＹ）は、ここまでの配置を成功とみなすことができ、一方、出力ノードＮ’が存在する場合（Ｓ４４のＮ）、出力ノード配置モードを再帰的に呼び出す（Ｓ４６）。

【００７８】

再帰的に呼び出した出力ノード配置モードでは、Ｓ３０において、１段下がった段にて、ノード６の配置可能な位置を探索する。この出力ノード配置モードでは、(0,1)に配置したノード２を基準として、ノード６の配置を定めていくことになる。以下では、座標(0,0)のノード１に対してノード２の座標を(0,1)に定めたように、ノード２に対して、ノード６の座標が(0,2)に定まる。この座標の決定は、図４０におけるＳ５６にて行われる。

【００７９】

続いてＳ５８にて、ノード６の入力ノード３が取り出される。なお、ノード６の入力ノードとしてノード２も存在するが、既に取り出したノードであるため、Ｓ５８では、新たに取り出すことはしない。ノード３が存在するため（Ｓ６０のＮ）、このノード３の配置をチェックするべく、段数を１段上げて（Y'←Y-1）、列を１つだけ左にずらす（X'←X-1）（Ｓ６２）。ここで、入力ノード配置モードを再帰的に呼び出し（Ｓ６４）、(X-1,Y-1)において、具体的には(-1,1)に、ノード３を配置可能であるか否かをチェックする。なお、X＜0（Ｓ５０のＹ）であるため、この座標(-1,1)にはノード３を配置することはできず（Ｓ６６のＮ）、X'を１インクリメントして（Ｓ６８）、X'が(X+1)を超えない限りにおいて（Ｓ７０のＮ）、隣の座標(0,1)においてノード３を配置することができるか否かをチェックする（Ｓ６４）。(0,1)には、既にノード２が配置されているため（Ｓ５２のＹ、Ｓ５４のＮ）、ノード３を配置することはできず（Ｓ６６のＮ）、さらに、X'を１インクリメントして（Ｓ６８）、(1,1)にノード３を配置することができるか否かをチェックする（Ｓ６４）。$0 \leqq X \leqq X_{MAX}$であり（Ｓ５０のＮ）、(1,1)が空いているため（Ｓ５２のＮ）、ノード３を(1,1)に配置する（Ｓ５６）。続いて、ノード３の入力ノードがノード１以外に存在しないため（Ｓ６０のＹ）、これまでの配置が成功したものとして、入力

ノード配置モードを抜ける。この状態は、図３０に示される。なお、Ｓ７０において、X'が(X+1)を超えた場合（Ｓ７０のＹ）、ＮおよびＮの入力を辿って消去する（Ｓ７２）。この場合、図３９のＳ３４に戻る。

【００８０】

続いて、入力ノード配置モード（Ｓ３２）において、配置が成功したため（Ｓ３４のＹ）、ノード配置チェックモード（Ｓ４０）を実行し、さらにＳ４２～Ｓ４６までのステップを実行する。これにより、これまでと同様に、ノード６の出力を入力にもつノード８を配置する。Ｓ５６において、ノード８は、(0,3)ないしは(0,1)に配置することができる。図３１は、ノード８を(0,3)に配置した状態を、図３２は、ノード８を(1,3)に配置した状態を示す。その後、Ｓ５８においてノード８の入力ノードであるノード７を取り出し、ノード７を(1,2)または(2,2)に配置し（Ｓ６４）、さらに、ノード７の入力ノードであるノード４を取り出して（Ｓ５８）、ノード４を(2,1)に配置することができるが、(2,1)に配置したノード４は、接続制限の関係から、(0,0)に配置したノード１とリンクすることができない。したがって、この配置は、失敗であることが分かる。

【００８１】

図３８のメインフローに戻って、以上のように、ノード１を(0,0)からスタートして、最も適切な配置状態を探索していく。最上段のノードN_0の配置を(0,0)に設定して探索処理を行い、最終的に出力ノードN_1がなくなるまで（Ｓ１８のＹ）、探索処理が完了すると、Ｓ２２において、X_0を１インクリメントして、ノード１の座標を(1,0)に設定して、探索をやり直す。なお、ノード１を(0,0)に配置したときに、全ノードを配置することができた場合は、その時点でメインフローを終了してもよい。なお、探索の繰り返しは、X_0がX_{MAX}以下の間行われ（Ｓ２４のＹ）、X_0がX_{MAX}より大きくとなると（Ｓ２４のＮ）、この探索処理は終了する。このように、各段に対して全探索を行う場合と比較すると、大幅に探索数を低減することが可能となる。また、できる限り多くのノードが１つのサブＤＦＧに含まれるように割り当てられるので、効率的な処理が可能となる。

【００８２】

（実施例８）

実施例８では、ＤＦＧの分割処理について説明する。

図４２は、図２５の各ノードにノード番号を割り振った状態を示す。まず、図２７（ａ）に示すように、図４２における左側のノード１～４をＡＬＵアレイに割り当て、ノード４の出力を入力にもつノード９の横位置を、ノード４の左下、真下、右下の順に探索を行う。それぞれの位置において、ノード９の上段にノード８を配置し、その入力であるノード５～７の配置を探索する。この探索処理は、実施例７で説明したとおりである。

【００８３】

この例では、ノード９及びその入力ノード５～７を配置する際、ノード５～７の全てを配置することができない。そこで、ノード９の配置が失敗した時点で、ノード９を除いた状態で一旦ＤＦＧを切り出し、残ったＤＦＧに対して、再度割り当てを行う。図２７（ｂ）が、残りのＤＦＧに対して割り当てたＡＬＵアレイを示す。このようにして、ＤＦＧを分割することが可能となる。分割したサブＤＦＧは、実施例７において図４１に関連して説明したノード配置チェックモードにおいて保存されたものを利用する。既述したように、Ｓ８４では、配置することができた最大数のノードの配置状態を保持するため、これをもとにサブＤＦＧ（ないしは、その構成情報）を形成することが可能となる。なお、この配置状態は、Ｓ８４において記憶部３４（図６参照）に記憶される。このような分割手法を用いることにより、できる限り多くのノードが１つのサブＤＦＧに含まれるように分割できるので、効率的な処理が可能となる。また、後述のサブＤＦＧの合成手法との組み合わせにより、全体としてサブＤＦＧの数を低減できる。

【００８４】

（実施例９）

次に、サブＤＦＧの合成について、例に沿って説明する。図４３は、分割して得られたサブＤＦＧの例を示す。なお、実際のＤＦＧは、図３や図１０などにおいて繰り返し示し

たように、ノードと接続関係を示した処理の流れを表現するものであるが、ここでは、説明の便宜上、ＡＬＵアレイにマッピングした状態のものと等価なものとして扱っている。図４３において、"dfgoutx-x"はサブＤＦＧの出力データを表す。例えば、サブＤＦＧ１の出力データdfgout1-1は、サブＤＦＧ２に入力され、サブＤＦＧ２の出力dfgout2-1、dfgout2-2は、それぞれサブＤＦＧ３、サブＤＦＧ５に入力される。分割後の状態では、ＡＬＵアレイの右側にノードが割り当てられていないＡＬＵが多いので、複数のサブＤＦＧを１つに合成して処理できれば、使用しているＡＬＵの並列度が向上し、高速化できる。但し、サブＤＦＧ間でデータの入出力が存在するので、そのような関係にあるサブＤＦＧは合成できない、そのため、まずサブＤＦＧ間で合成可能か否かを判定し、合成可能であれば、実際の合成処理を行う。サブＤＦＧ間で合成可能か否かの判定は、まず、サブＤＦＧを１つのノードとし、その入出力データの流れをデータフローとしてＤＦＧを作成することで行ってもよい。

【００８５】

図６に戻って、ＤＦＧ生成部３０は、同時実行可能な複数の回路に対応するサブＤＦＧを探索して合成し、新たなＤＦＧを生成する。合成可能なサブＤＦＧは、入出力において互いに独立していることが条件となる。構成情報生成部３２は、合成したＤＦＧから構成情報を生成し、記憶部３４に記憶させる。制御部１８は、この構成情報を設定部１４を介してリコンフィギュラブル回路１２に供給する。これにより、リコンフィギュラブル回路１２には、同時実行可能な複数の回路が同時に形成されることになる。

【００８６】

図４４は、サブＤＦＧ間のデータフローを示す。ＤＦＧ生成部３０は、図４４のデータフローをもとに、合成可能なサブＤＦＧを探索する。サブＤＦＧ１の出力を直接的または間接的に受け取るサブＤＦＧ２、サブＤＦＧ３およびサブＤＦＧ５は、サブＤＦＧ１と合成することはできないが、サブＤＦＧ１とサブＤＦＧ４は入出力データを受け取る関係にはないため、合成しても問題はない。

【００８７】

図４５は、合成判定を行うフローチャートを示す。まず、ｉを１にセットする（Ｓ１００）。ｉがサブＤＦＧの個数以下であれば（Ｓ１０２のＹ）、ｉ番目のサブＤＦＧｉを取り出す（Ｓ１０４）。このサブＤＦＧｉについて、入力ＤＦＧチェックモード（Ｓ１０６）および出力ＤＦＧチェックモード（Ｓ１０８）における処理を実行する。

【００８８】

図４６は、入力ＤＦＧチェックモードにおける処理のフローチャートを示す。まず、サブＤＦＧｉの入力ＤＦＧを取り出す（Ｓ１２０）。取り出したＤＦＧを、以下のフローでは、ＤＦＧｉとして定義する。ＤＦＧｉが存在しない場合（Ｓ１２２のＹ）、Ｓ１０４で取り出したサブＤＦＧｉの入力が存在しないことが判定され、入力ＤＦＧチェックモードにおける処理を終了する。一方、入力ＤＦＧｉが存在する場合（Ｓ１２２のＮ）、そのＤＦＧｉをチェックして（Ｓ１２４）、再度、入力ＤＦＧモードにおける処理を再帰的に実行する（Ｓ１２６）。この再帰処理は、入力ＤＦＧｉが存在しなくなった時点で（Ｓ１２２のＹ）で、終了する。

【００８９】

図４７は、出力ＤＦＧチェックモードにおける処理のフローチャートを示す。まず、サブＤＦＧｉの出力ＤＦＧを取り出す（Ｓ１３０）。取り出したＤＦＧを、以下のフローでは、ＤＦＧｉとして定義する。ＤＦＧｉが存在しない場合（Ｓ１３２のＹ）、Ｓ１０４で取り出したサブＤＦＧｉの出力が存在しないことが判定され、出力ＤＦＧチェックモードにおける処理を終了する。一方、出力ＤＦＧｉが存在する場合（Ｓ１３２のＮ）、そのＤＦＧｉをチェックして（Ｓ１３４）、再度、出力ＤＦＧモードにおける処理を再帰的に実行する（Ｓ１３６）。この再帰処理は、出力ＤＦＧｉが存在しなくなった時点で（Ｓ１３２のＹ）で、終了する。

【００９０】

図４５に戻って、ＤＦＧｉは、未チェックのサブＤＦＧと合成可能であることが判明す

る（Ｓ１１０）。すなわち、入力ＤＦＧチェックモードおよび出力ＤＦＧチェックモードにおいてチェックされなかったサブＤＦＧは、サブＤＦＧｉとの間で、互いの処理に影響を及ぼしあわないことが分かる。つまり、サブＤＦＧｉは、未チェックのサブＤＦＧに入力データを直接的または間接的に供給するものではなく、また未チェックのサブＤＦＧからの出力データを直接的または間接的に必要とするものではない。合成可能なサブＤＦＧは、サブＤＦＧｉと対応付けて記憶される。続いて、チェックを解除し（Ｓ１１２）、ｉを１インクリメントして（Ｓ１１４）、再度、合成判定の処理を実行する。ｉがサブＤＦＧの個数を超えると（Ｓ１０２のＮ）、この合成判定処理を終了する。

【００９１】

以上のフローを用いると、図４４の例では、サブＤＦＧ１とサブＤＦＧ４の合成が可能であることが分かる。さらに、図４４の例では、サブＤＦＧ２とサブＤＦＧ４の組合せ、サブＤＦＧ３とサブＤＦＧ４の組合せ、さらにはサブＤＦＧ３とサブＤＦＧ５の組合せが可能であることが分かる。このように、複数のサブＤＦＧを１つのＤＦＧに合成することができるので、並列度が増し、高速化が可能となる。

【００９２】

以上、複数の実施例をもとに本発明を説明した。なお本発明はこの実施例に限定されることなく、そのさまざまな変形例もまた、本発明の態様として有効である。特に、本発明に関して、実施例１から９まで例示したが、これらは単独で用いられてもよいが、それぞれを組み合わせたものも、本発明の範囲内にある。例えば、上記した実施例において、リコンフィギュラブル回路１２における論理回路の機能を決定する方法を説明したが、この機能決定方法は、リコンフィギュラブル回路に限らず、他の回路、例えば固定ハードウェアの回路設計に利用することも可能である。回路実装後の固定ハードウェアにおいては、ＤＦＧを複数に分割して回路を再生成するなどの処理は行わないが、素子形成の前段階となる回路設計を行なう段階に本手法を利用することで、効率よく回路素子の機能および配置を決定して、決定した位置に決定した機能をもつ回路素子を形成することで、所期の回路を生成することが可能となる。リコンフィギュラブル回路１２の回路機能を決定する場合には、複数のＡＬＵにＤＦＧのノードを対応付ける処理を行っていたが、固定ハードウェアの回路を生成する場合には、ＡＬＵアレイが存在するわけではないため、基板上の回路素子を配置する位置、例えば他の回路素子との並びにおける順番なども決めることになる。その他の処理は、上記した実施例と同様である。その結果、所期の回路を簡易に生成することが可能となる。

【図面の簡単な説明】

【００９３】

【図１】ＡＬＵアレイを用いたリコンフィギュラブルプロセッサの構成図である。

【図２】Ｃ言語プログラムの一例を示す図である。

【図３】図２に示すプログラムに対応するＤＦＧを示す図である。

【図４】図３のＤＦＧから構成情報をＡＬＵアレイに割り当てた時の構成図である。

【図５】縦方向の段をまたぐＡＬＵ間接続をなくした接続方式を示す図である。

【図６】実施例２に係る処理装置の構成図である。

【図７】１つの生成するべき回路を分割してできる複数の回路の構成情報について説明するための図である。

【図８】実施例２による接続方式のリコンフィギュラブル回路１２の構成図である。

【図９】図２に示すＣプログラムを図８に示したＡＬＵアレイに割り当てた結果を示す図である。

【図１０】データフローグラフの例を示す図である。

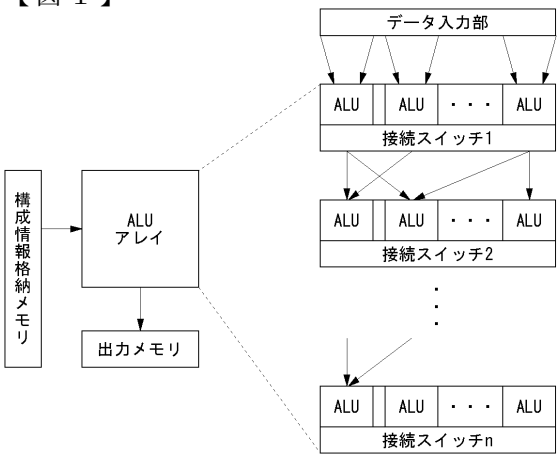
【図１１】信号処理のフローチャートである。

【図１２】ＡＬＵアレイの縦方向のサイズを超えたＤＦＧをＡＬＵアレイにマッピングした仮想的な状態の一例を示す図である。

【図１３】図１２におけるＤＦＧを２つに分割したサブＤＦＧをＡＬＵアレイにマッピングした状態を示す図である。

【図１４】縦横の両方向にＡＬＵアレイのサイズを超えるＤＦＧをＡＬＵアレイにマッピングした仮想的な状態の一例を示す図である。
【図１５】図１４におけるＤＦＧを分割したそれぞれのサブＤＦＧをＡＬＵアレイにマッピングした状態を示す図である。
【図１６】Ｃプログラムの例を示す図である。
【図１７】図１６に示すＣプログラムのＤＦＧを示す図である。
【図１８】ノードコピーを行って、出力数を３つに制限したＤＦＧを示す図である。
【図１９】図１８に示したＤＦＧを割り当てたＡＬＵアレイを示す図である。
【図２０】スルーノードを挿入したＤＦＧを示す図である。
【図２１】図２０に示したＤＦＧを割り当てたＡＬＵアレイを示す図である。
【図２２】分割した２つのサブＤＦＧを示す図である。
【図２３】図２２に示したサブＤＦＧを割り当てたＡＬＵアレイを示す図である。
【図２４】条件文を含んだＣプログラムの一例を示す図である。
【図２５】図２４のＣプログラムに対応したＤＦＧを示す図である。
【図２６】図２５のＤＦＧをそのまま割り当てたＡＬＵアレイを示す図である。
【図２７】生成したサブＤＦＧを割り当てたＡＬＵアレイを示す図である。
【図２８】図１８に示すＤＦＧの各ノードにノード番号を割り振った状態を示す図である。
【図２９】ノード１を配置した状態を示す図である。
【図３０】複数のノードを配置した状態を示す図である。
【図３１】ノード８を(0,3)に置いた図である。
【図３２】ノード８を(1,3)に置いた図である。
【図３３】ノード６を(1,2)に置いた図である。
【図３４】ノード１を(1,0)に配置して、ノード２、ノード６、ノード３の順にそれぞれの配置を決定した状態を示す図である。
【図３５】ノード６に対して、ノード８、ノード７、ノード４の配置を決定した状態を示す図である。
【図３６】ノード７を(2,2)に置いた図である。
【図３７】ＡＬＵアレイにおける全てのノード配置が決定された状態を示す図である。
【図３８】ノード配置のメインフローチャートである。
【図３９】出力ノード配置モードの処理フローチャートである。
【図４０】入力ノード配置モードの処理フローチャートである。
【図４１】ノード配置チェックモードの処理フローチャートである。
【図４２】図２５の各ノードにノード番号を割り振った状態を示す図である。
【図４３】分割して得られたサブＤＦＧの例を示す図である。
【図４４】サブＤＦＧ間のデータフローを示す図である。
【図４５】合成判定を行うフローチャートである。
【図４６】入力ＤＦＧチェックモードにおける処理のフローチャートである。
【図４７】出力ＤＦＧチェックモードにおける処理のフローチャートである。
【符号の説明】
【００９４】
１０・・・処理装置、１２・・・リコンフィギュラブル回路、１４・・・設定部、１６・・・選択器、１８・・・制御部、２０・・・内部状態保持回路、２２・・・出力回路、２４・・・経路部、２６・・・集積回路装置、３０・・・ＤＦＧ生成部、３２・・・構成情報生成部、３４・・・記憶部、３８・・・データフローグラフ、４０・・・構成情報、５０・・・論理回路。

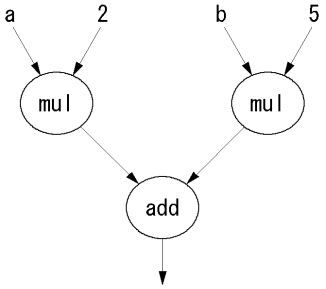
【図１】



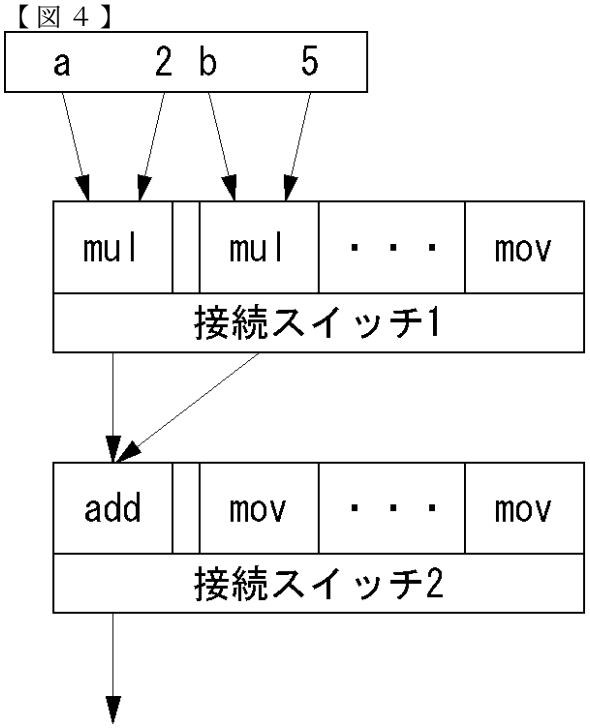
【図２】

```
int  main(int a, int b){
    int x = 2 * a;
    int y = 5 * b;
    return (x + y);
}
```

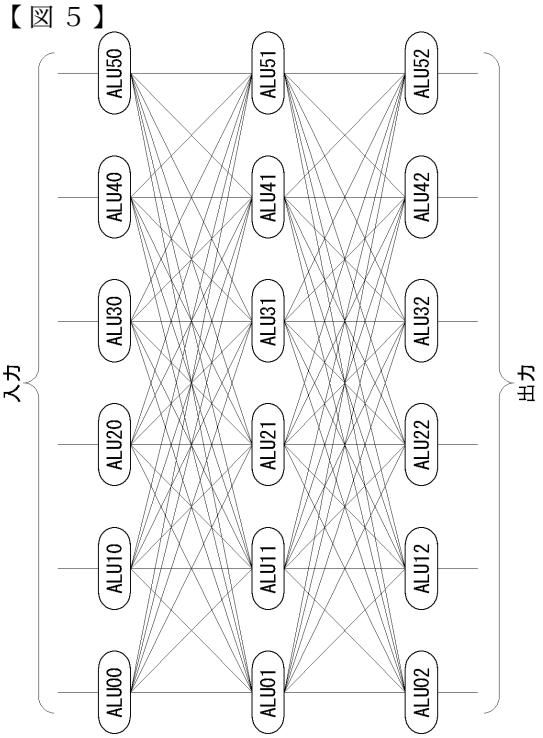
【図３】



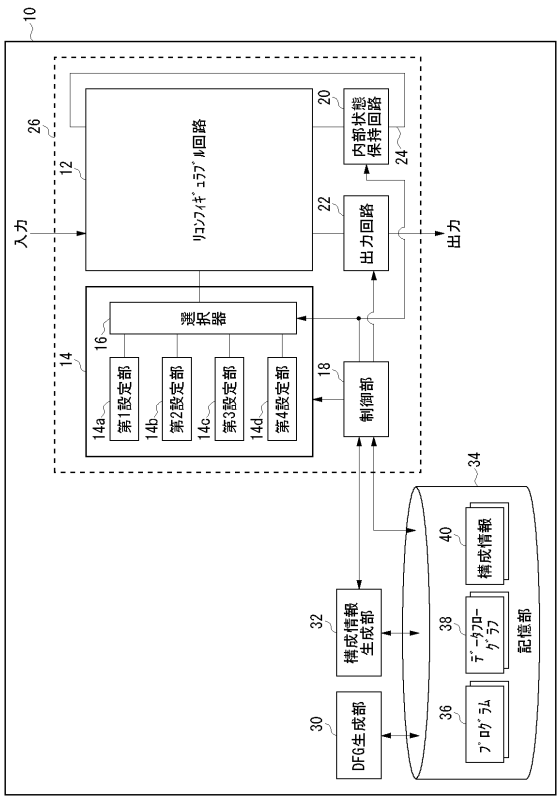
【図４】



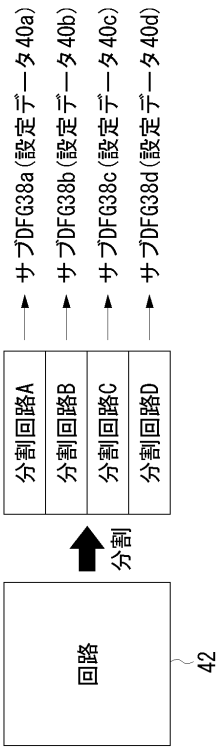
【図５】



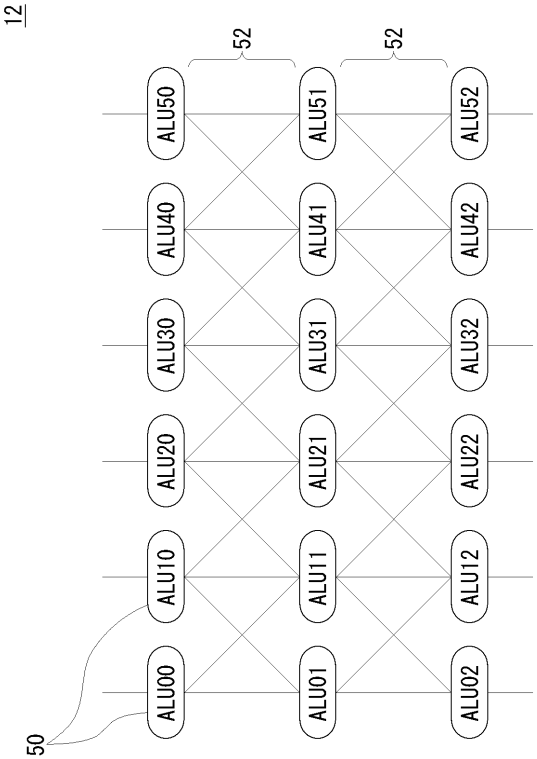
【図６】


10
入力
出力
リコンフィギュラブル回路
選択器
第1設定部
第2設定部
第3設定部
第4設定部
制御部
出力回路
内部状態保持回路
構成情報生成部
DFG生成部
構成情報
データ
プログラム
記憶部

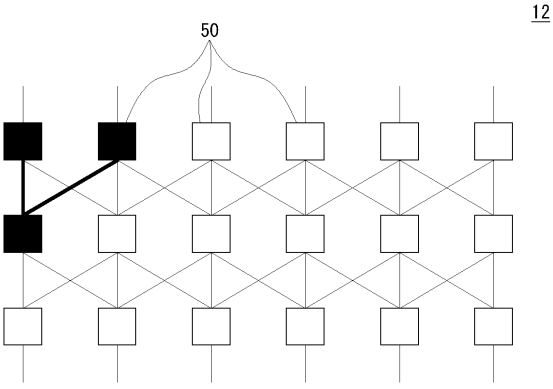
【図７】


サブDFG38a（設定データ40a）
サブDFG38b（設定データ40b）
サブDFG38c（設定データ40c）
サブDFG38d（設定データ40d）
分割回路A
分割回路B
分割回路C
分割回路D
分割
回路
42

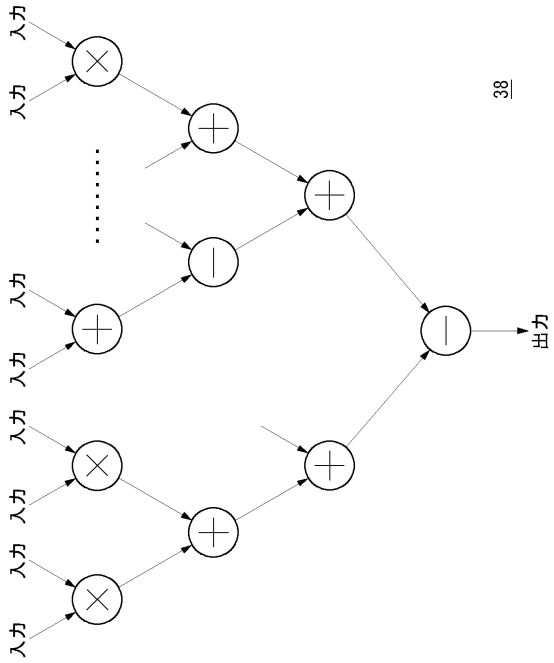
【図８】


12
52
52
ALU50
ALU51
ALU52
ALU40
ALU41
ALU42
ALU30
ALU31
ALU32
ALU20
ALU21
ALU22
ALU10
ALU11
ALU12
ALU00
ALU01
ALU02
50

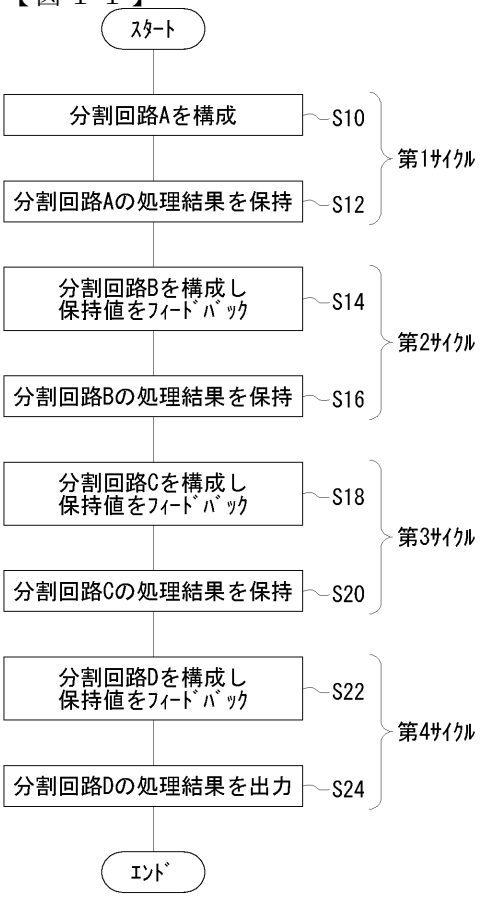
【図９】


12
50

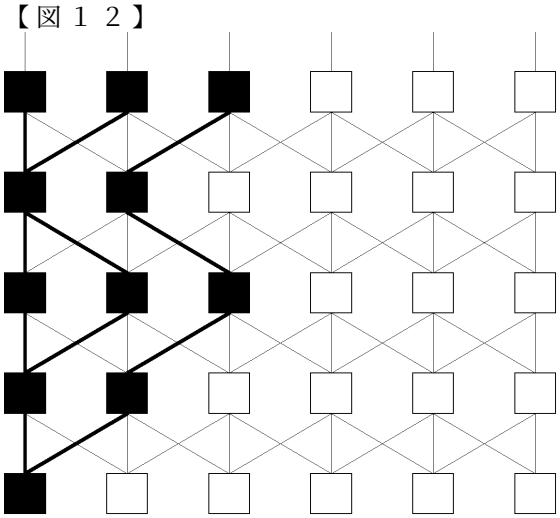
【図１０】



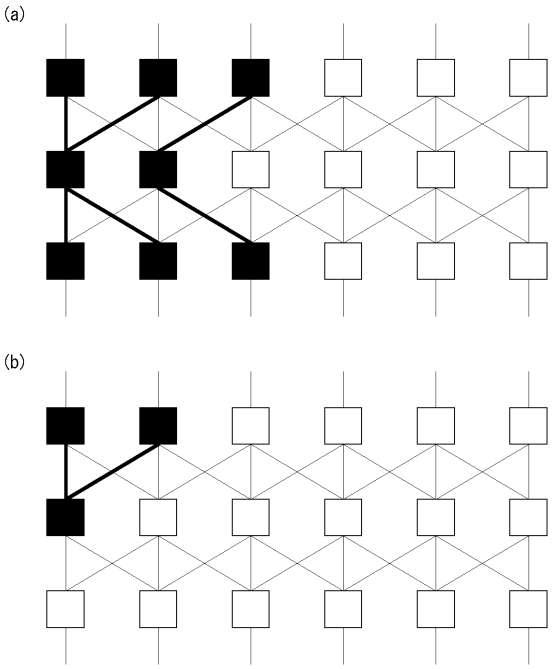
【図１１】



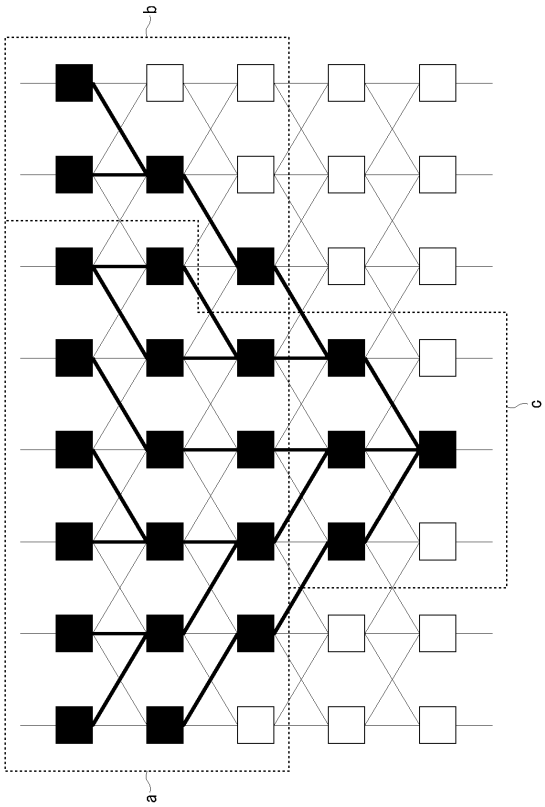
【図１２】



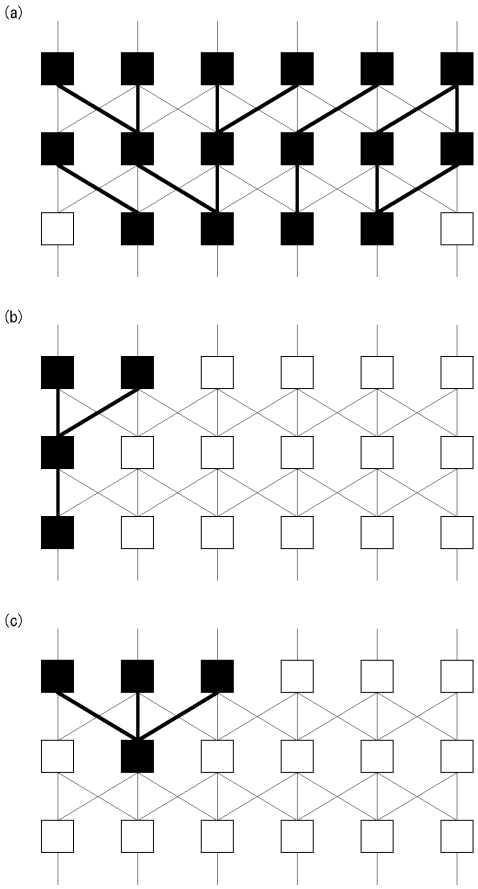
【図１３】



【図１４】



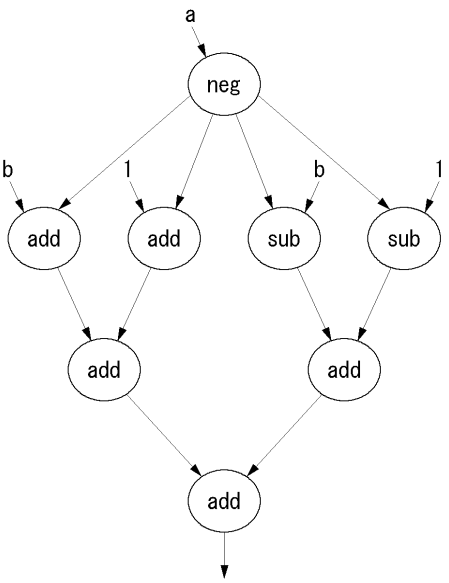
【図１５】



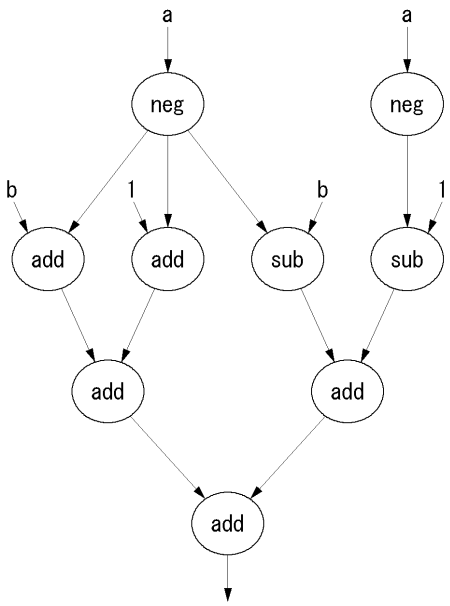
【図１６】

```
int  main(int a, int b){
    int r1, r2, r3, r4, tmp;
    tmp = -a;
    r1 = tmp + 1;
    r2 = tmp + b;
    r3 = tmp - 1;
    r4 = tmp - b;
    return (r1 + r2) + (r3 + r4);
}
```

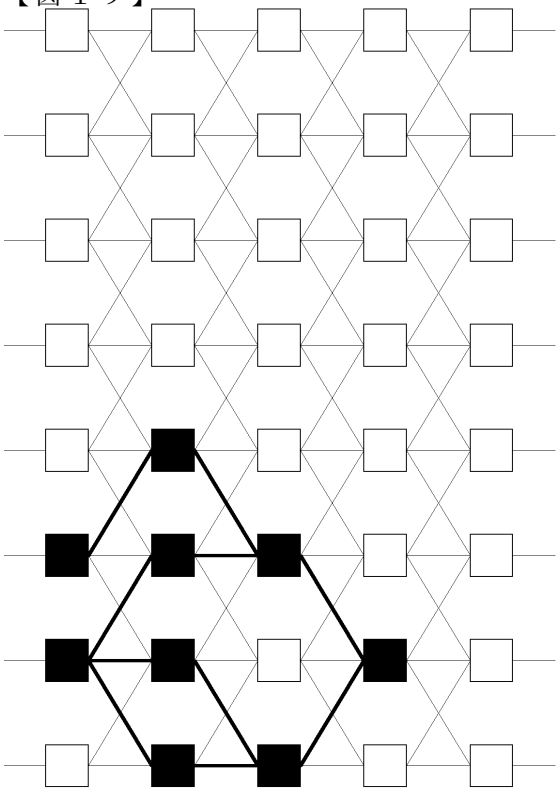
【図１７】



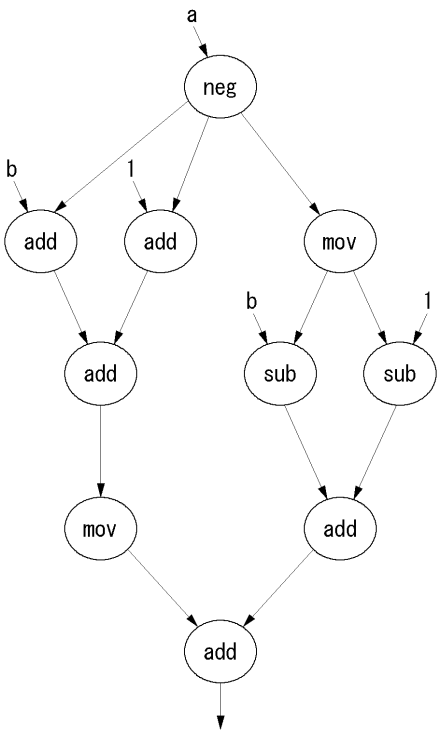
【図１８】



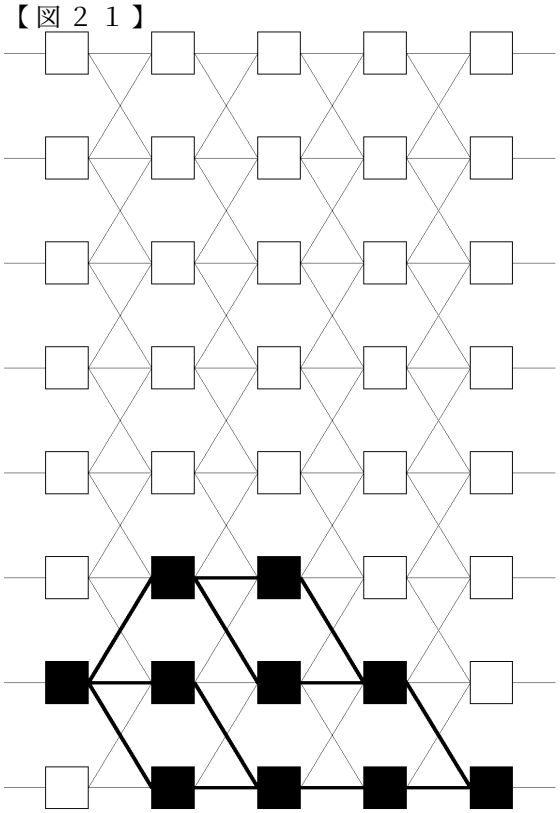
【図１９】



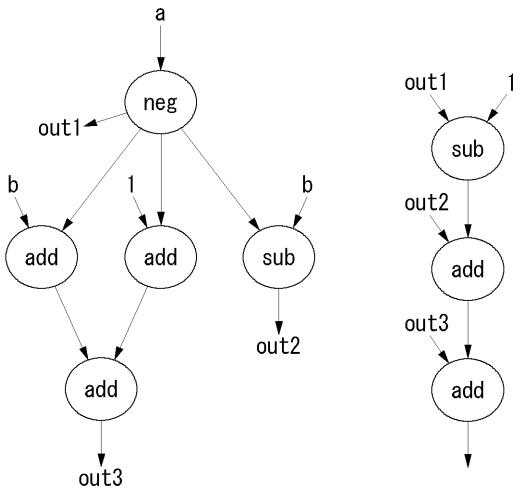
【図２０】



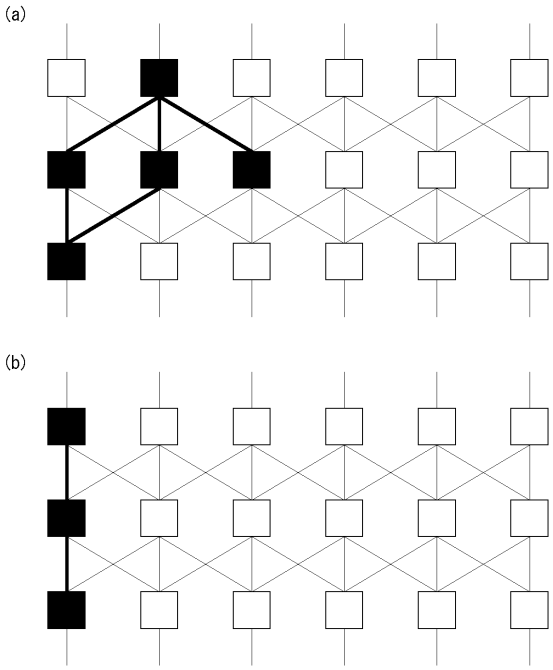
【図２１】



【図２２】



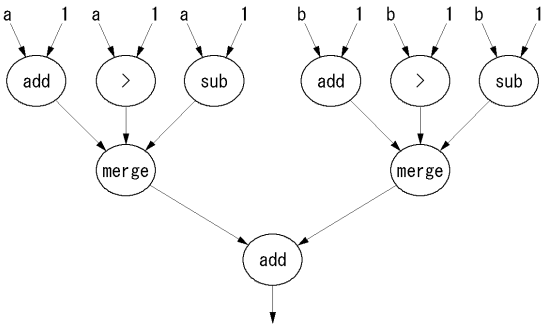
【図２３】



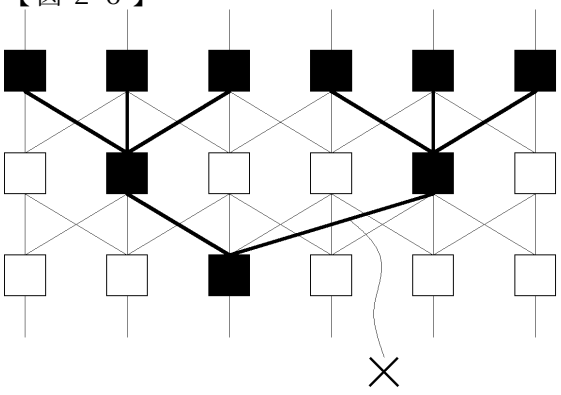
【図２４】

```
int  main(int a, int b){
  int x, y;
  if(a > 1)   x = a + 1;
  else        x = a - 1;
  if(b > 1)   y = b + 1;
  else        y = b - 1;
  return x + y;
}
```

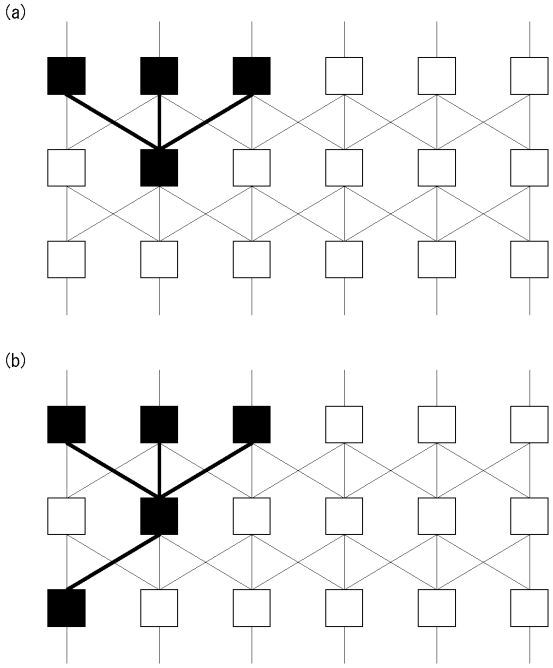
【図２５】



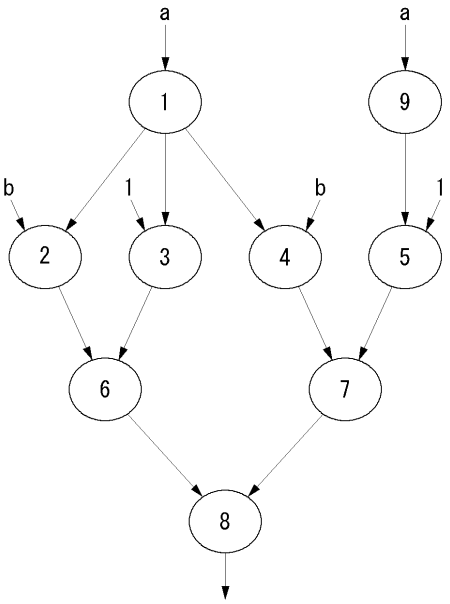
【図２６】



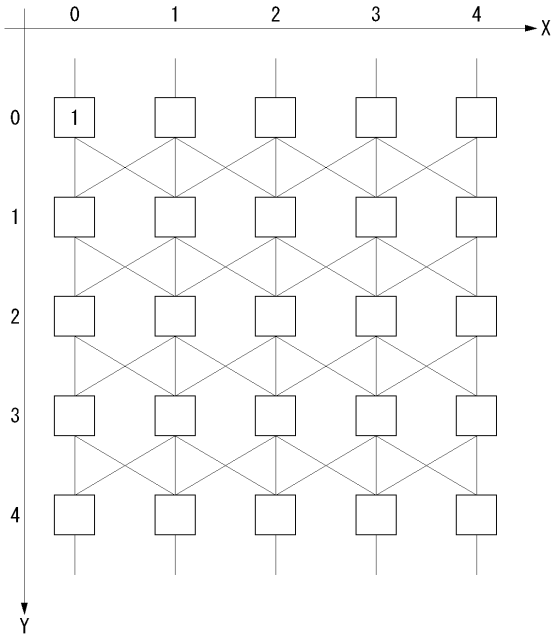
【図２７】



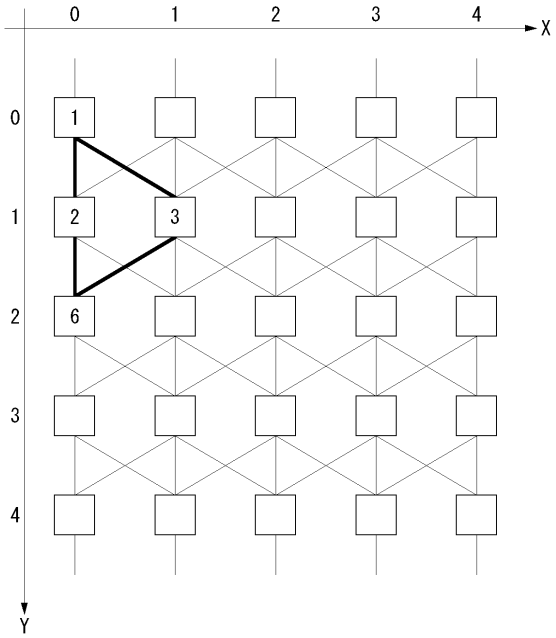
【図２８】



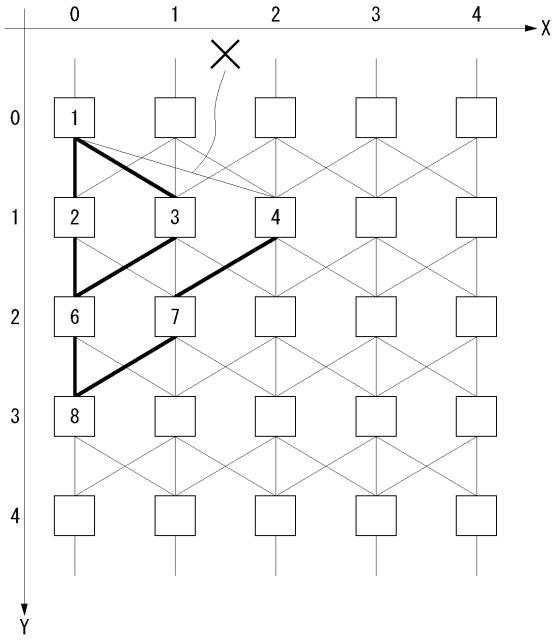
【図２９】



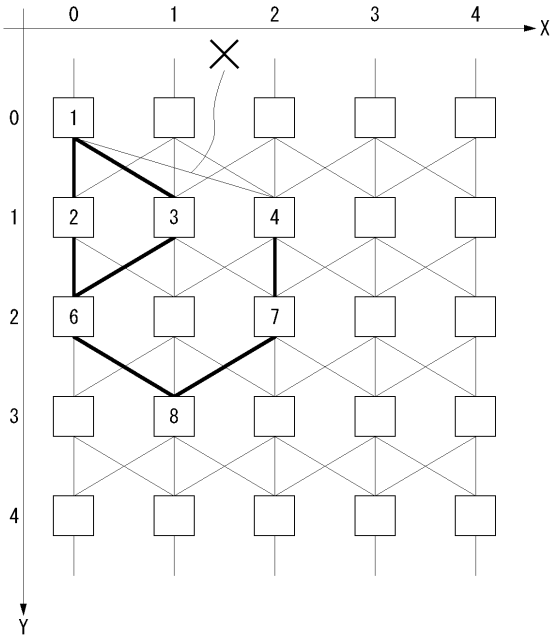
【図３０】



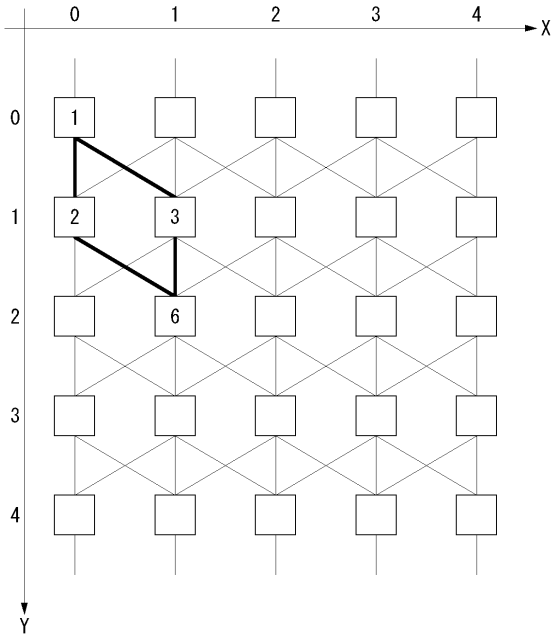
【図３１】



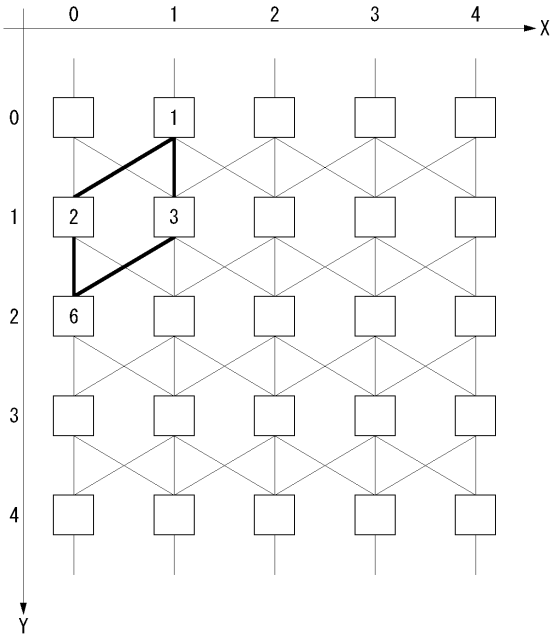
【図３２】



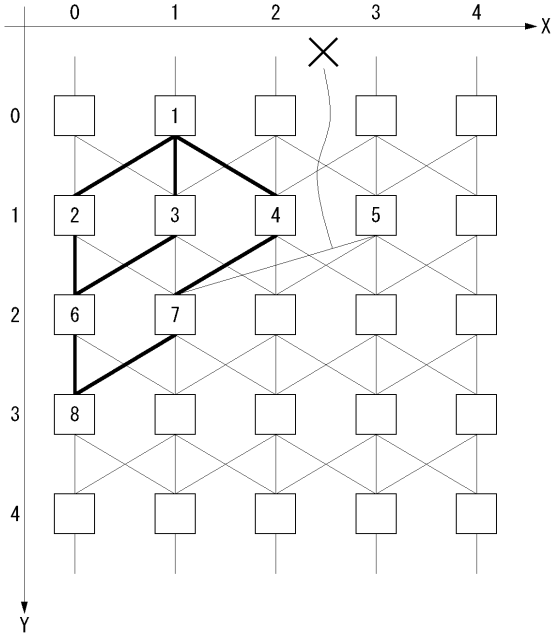
【図３３】



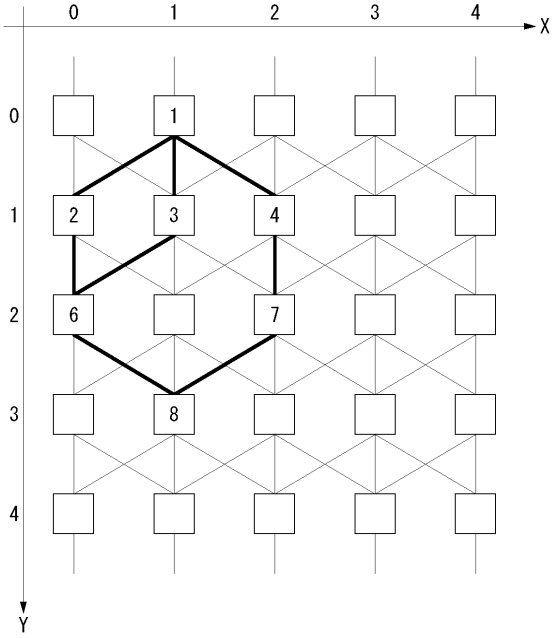
【図３４】



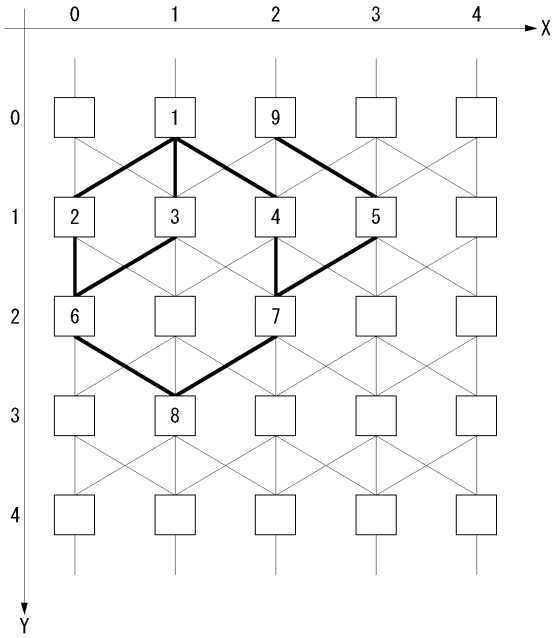
【図３５】



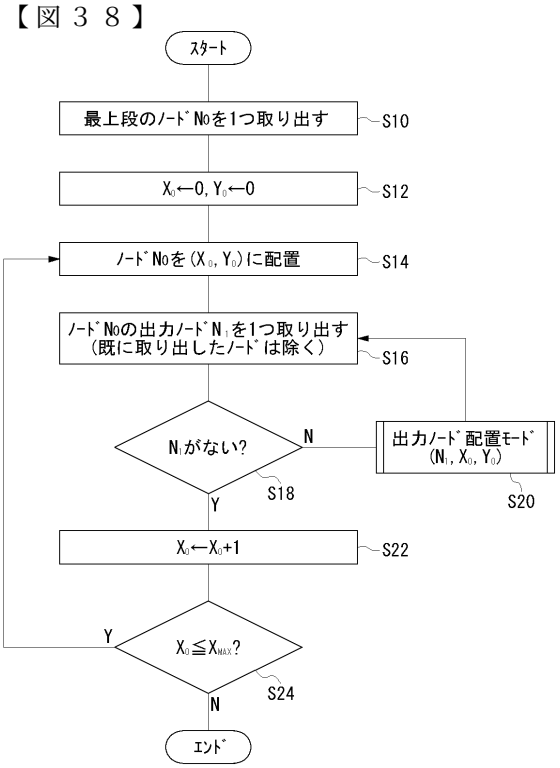
【図３６】



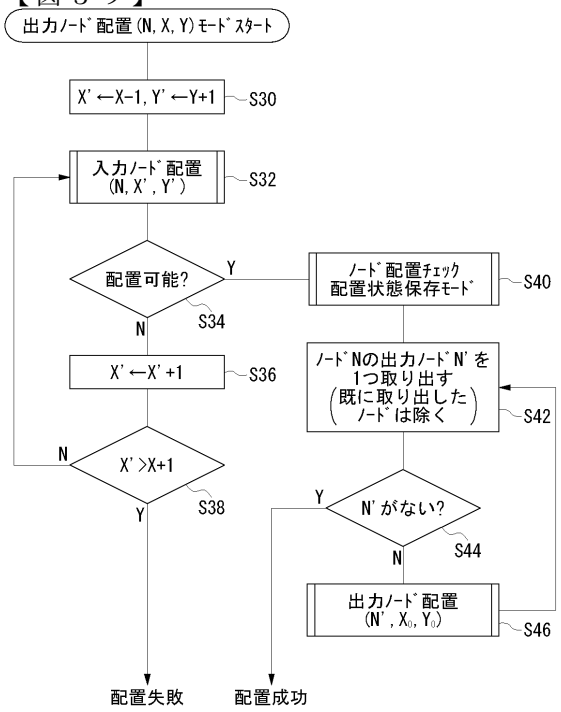
【図３７】



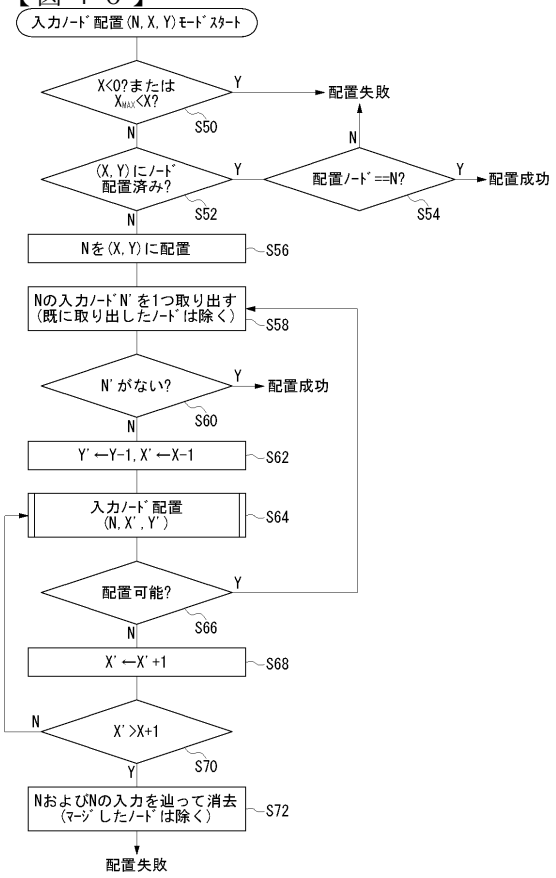
【図３８】



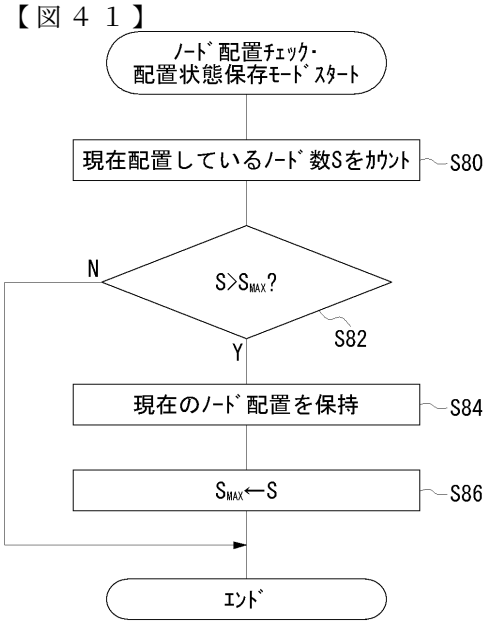
【図３９】



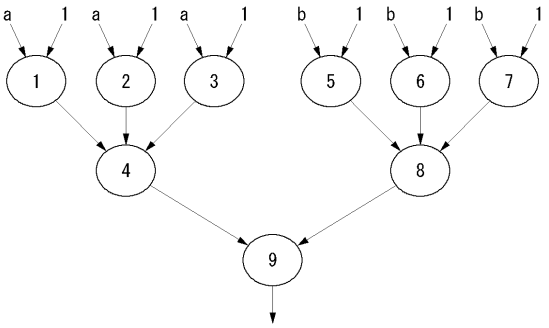
【図４０】



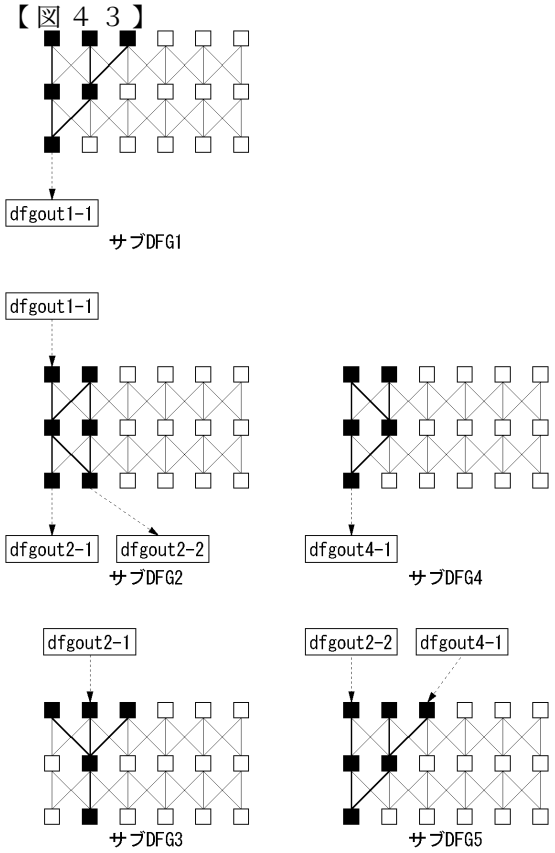
【図４１】



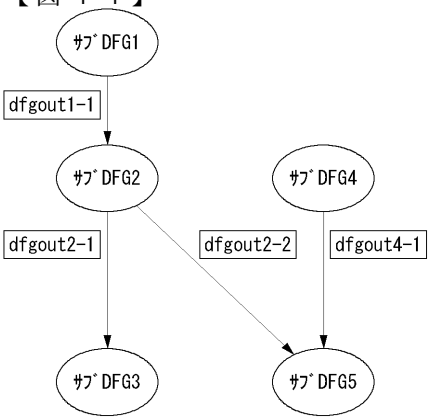
【図４２】



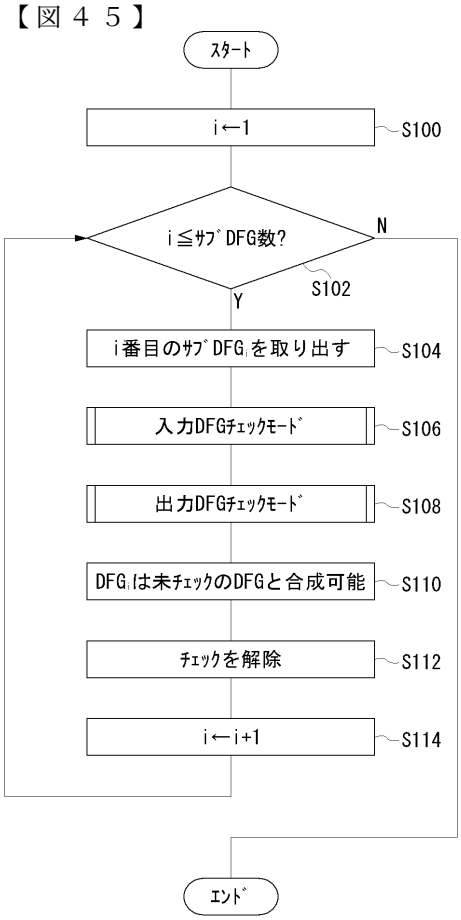
【図４３】



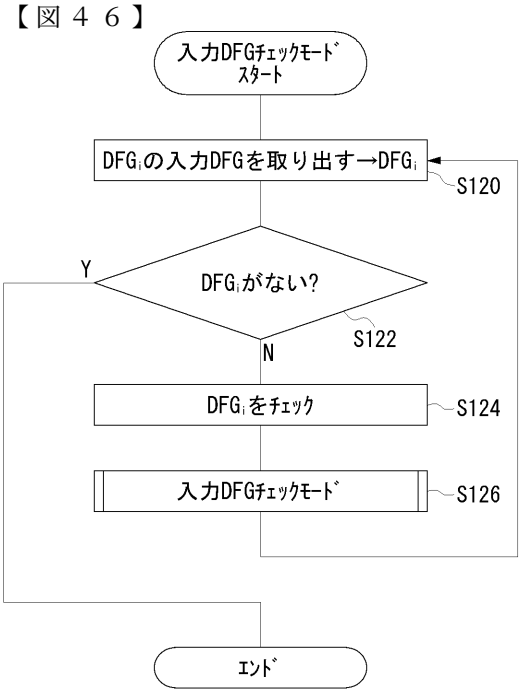
【図４４】



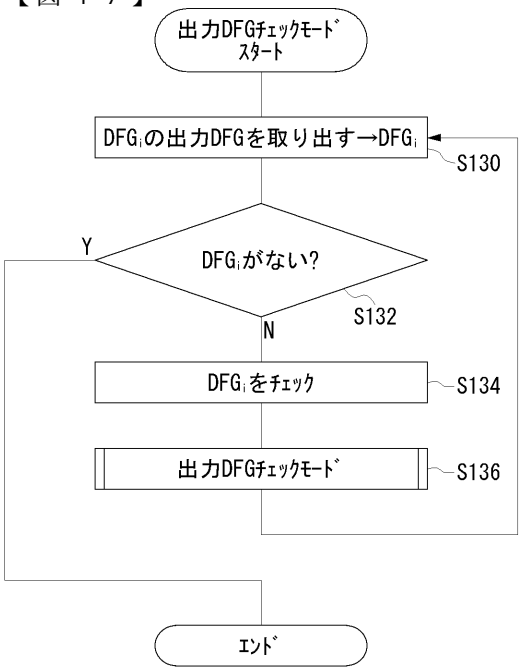
【図４５】



【図４６】



【図４７】


出力DFGﾁｪｯｸﾓｰﾄﾞ
ｽﾀｰﾄ
DFGiの出力DFGを取り出す→DFGi
S130
Y
DFGiがない?
S132
N
DFGiをﾁｪｯｸ
S134
出力DFGﾁｪｯｸﾓｰﾄﾞ
S136
ｴﾝﾄﾞ

フロントページの続き

(51)Int.Cl.[7]	FI	テーマコード（参考）
	Ｈ０３Ｋ 19/173 １０１	

(72)発明者 小曽根 真
大阪府守口市京阪本通２丁目５番５号 三洋電機株式会社内

Ｆターム(参考) 5B022 AA07 BA00 CA03 FA03
5B046 AA08 BA03
5F064 AA08 BB02 DD19 EE15 FF04 FF52 HH01 HH06 HH11
5J042 BA01 BA11 CA02 CA16 CA19 CA20 DA01