(19)日本国特許庁(JP)　(12)公表特許公報(A)　(11)公表番号
特表2022-534068
(P2022-534068A)
(43)公表日　令和4年7月27日(2022.7.27)

(51)国際特許分類		FI			テーマコード（参考）
G06F 30/34 (2020.01)		G06F	30/34		5B045
G06F 15/78 (2006.01)		G06F	15/78	530	5B062
G06F 15/80 (2006.01)		G06F	15/78	560	5B146
G06F 15/173 (2006.01)		G06F	15/80		
G06F 30/347 (2020.01)		G06F	15/173	681	

審査請求　未請求　予備審査請求　未請求　（全39頁）　最終頁に続く

(21)出願番号　特願2021-569551(P2021-569551)
(86)(22)出願日　令和2年5月7日(2020.5.7)
(85)翻訳文提出日　令和3年12月22日(2021.12.22)
(86)国際出願番号　PCT/US2020/031951
(87)国際公開番号　WO2020/236436
(87)国際公開日　令和2年11月26日(2020.11.26)
(31)優先権主張番号　16/420,881
(32)優先日　令和1年5月23日(2019.5.23)
(33)優先権主張国・地域又は機関
米国(US)
(81)指定国・地域　AP(BW,GH,GM,KE,LR,LS,MW,MZ,NA,RW,SD,SL,ST,SZ,TZ,UG,ZM,ZW),EA(AM,AZ,BY,KG,KZ,RU,TJ,TM),EP(AL,AT,BE,BG,CH,CY,CZ,DE,DK,EE,ES,FI,FR,GB,GR,HR,HU,IE,IS,IT,LT,LU,LV,MC,
最終頁に続く

(71)出願人　591025439
ザイリンクス インコーポレイテッド
ＸＩＬＩＮＸ ＩＮＣＯＲＰＯＲＡＴＥＤ
アメリカ合衆国 カリフォルニア州 ９５１２４－３４００ サン ホセ ロジック ドライブ ２１００
(74)代理人　110001195弁理士法人深見特許事務所
(72)発明者　シバラマン，ムクンド
アメリカ合衆国、９５１２４ カリフォルニア州、サン・ノゼ、ロジック・ドライブ、２１００
(72)発明者　グプタ，シャイル・アディティア
アメリカ合衆国、９５１２４ カリフォルニア州、サン・ノゼ、ロジック・ドライブ、２１００
最終頁に続く

(54)【発明の名称】　異種マルチコアアーキテクチャのコンパイルフロー

(57)【要約】
データ処理エンジン（ＤＰＥ）アレイを有するシステムオンチップ（ＳＯＣ）のアプリケーションを実装する例示的な方法は、アプリケーションのグラフ表現を決定するステップであって、グラフ表現は、アプリケーションのカーネルを表すノード、および、カーネル間の通信を表すエッジを含む、ステップと、グラフに基づいて、カーネルをＤＰＥアレイのＤＰＥにマッピングし、カーネルのデータ構造をＤＰＥアレイ内のメモリにマッピングするステップと、ＤＰＥとＳＯＣのプログラマブルロジック内に構成されたアプリケーションの回路との間の通信チャネルをルーティングするステップと、マッピングおよびルーティングの結果に基づいてアプリケーションを実装するためにＳＯＣをプログラミングするための実装データを生成するステップとを含む。

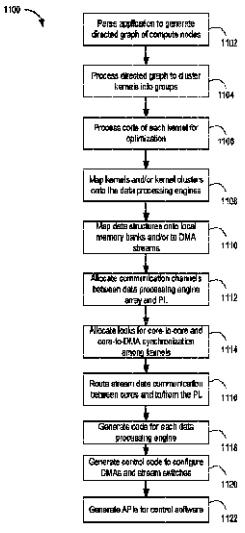


FIG. 11

【特許請求の範囲】
【請求項１】
データ処理エンジン（ＤＰＥ）アレイを有するシステムオンチップ（ＳＯＣ）のアプリケーションを実装する方法であって、
前記アプリケーションのグラフ表現を決定するステップであって、前記グラフ表現は、前記アプリケーションのカーネルを表すノード、および、前記カーネル間の通信を表すエッジを含む、グラフ表現を決定するステップと、
前記グラフに基づいて、前記カーネルを前記ＤＰＥアレイのＤＰＥにマッピングし、前記カーネルのデータ構造を前記ＤＰＥアレイ内のメモリにマッピングするステップと、
ＤＰＥと前記ＳＯＣのプログラマブルロジック内に構成された前記アプリケーションの回路との間の通信チャネルをルーティングするステップと、
前記マッピングおよび前記ルーティングの結果に基づいて前記アプリケーションを実装するために前記ＳＯＣをプログラミングするための実装データを生成するステップと
を含む、方法。
【請求項２】
前記マッピングするステップの前に、グラフ表現を処理してカーネルをグループにクラスタ化するステップをさらに含み、
前記マッピングするステップは、カーネルの少なくとも１つのクラスタを前記ＤＰＥアレイ内の１つのＤＰＥにマッピングするステップを含む、請求項１に記載の方法。
【請求項３】
前記マッピングするステップの前に、前記カーネルのうちの１つまたは複数のコードを最適化するステップと、
前記ＤＰＥアレイにおけるＤＰＥ－ＤＰＥ間通信のためにロックを配分するステップと
を含む、請求項１に記載の方法。
【請求項４】
前記マッピングするステップは、
前記データ構造を、前記ＤＰＥ内のメモリバンクおよびＤＰＥ間のダイレクトメモリアクセス（ＤＭＡ）ストリームのうちの少なくとも１つにマッピングするステップを含む、請求項１に記載の方法。
【請求項５】
プロセッサによって実行されると、データ処理エンジン（ＤＰＥ）アレイを有するシステムオンチップ（ＳＯＣ）のアプリケーションを実装する方法を前記プロセッサに実行させる命令を格納している非一時的コンピュータ可読媒体であって、前記方法は、
前記アプリケーションのグラフ表現を決定するステップであって、前記グラフ表現は、前記アプリケーションのカーネルを表すノード、および、前記カーネル間の通信を表すエッジを含む、ステップと、
前記グラフに基づいて、前記カーネルを前記ＤＰＥアレイのＤＰＥにマッピングし、前記カーネルのデータ構造を前記ＤＰＥアレイ内のメモリにマッピングするステップと、
ＤＰＥと前記ＳＯＣのプログラマブルロジック内に構成された前記アプリケーションの回路との間の通信チャネルをルーティングするステップと、
前記マッピングおよび前記ルーティングの結果に基づいて前記アプリケーションを実装するために前記ＳＯＣをプログラミングするための実装データを生成するステップと
を含む、非一時的コンピュータ可読媒体。
【請求項６】
前記マッピングするステップの前に、グラフ表現を処理してカーネルをグループにクラスタ化するステップをさらに含み、
前記マッピングするステップは、カーネルの少なくとも１つのクラスタを前記ＤＰＥアレイ内の１つのＤＰＥにマッピングするステップを含む、請求項５に記載の非一時的コンピュータ可読媒体。
【請求項７】

前記マッピングするステップの前に、前記カーネルのうちの１つまたは複数のコードを最適化するステップをさらに含む、請求項５に記載の非一時的コンピュータ可読媒体。
【請求項８】
前記マッピングするステップは、
前記データ構造を、前記ＤＰＥ内のメモリバンクおよびＤＰＥ間のダイレクトメモリアクセス（ＤＭＡ）ストリームのうちの少なくとも１つにマッピングするステップを含む、請求項５に記載の非一時的コンピュータ可読媒体。
【請求項９】
前記ＤＰＥアレイにおけるＤＰＥ－ＤＰＥ間通信のためにロックを配分するステップをさらに含む、請求項５に記載の非一時的コンピュータ可読媒体。
【請求項１０】
前記実装データを生成する前記ステップは、
前記カーネルのうちの１つまたは複数を実装するための各ＤＰＥのためのコードを生成するステップを含む、請求項５に記載の非一時的コンピュータ可読媒体。
【請求項１１】
前記実装データを生成する前記ステップは、
前記ＤＰＥアレイ内のダイレクトメモリアクセス（ＤＭＡ）およびスイッチ回路を構成するための制御コードを生成するステップを含む、請求項５に記載の非一時的コンピュータ可読媒体。
【請求項１２】
コンピュータシステムであって、
プログラムコードを格納するように構成されたメモリと、
データ処理エンジン（ＤＰＥ）アレイを有するシステムオンチップ（ＳＯＣ）のアプリケーションを実装するために、前記プログラムコードを実行するように構成されたプロセッサとを備え、前記プロセッサは、
前記アプリケーションのグラフ表現を決定するステップであって、前記グラフ表現は、前記アプリケーションのカーネルを表すノード、および、前記カーネル間の通信を表すエッジを含む、ステップと、
前記グラフに基づいて、前記カーネルを前記ＤＰＥアレイのＤＰＥにマッピングし、前記カーネルのデータ構造を前記ＤＰＥアレイ内のメモリにマッピングするステップと、
ＤＰＥと前記ＳＯＣのプログラマブルロジック内に構成された前記アプリケーションの回路との間の通信チャネルをルーティングするステップと、
前記マッピングおよび前記ルーティングの結果に基づいて前記アプリケーションを実装するために前記ＳＯＣをプログラミングするための実装データを生成するステップと、
を実行する、コンピュータシステム。
【請求項１３】
前記マッピングするステップの前に、グラフ表現を処理してカーネルをグループにクラスタ化するステップをさらに含み、
前記マッピングするステップは、カーネルの少なくとも１つのクラスタを前記ＤＰＥアレイ内の１つのＤＰＥにマッピングするステップ、または、前記データ構造を前記ＤＰＥ内のメモリバンクおよびＤＰＥ間のダイレクトメモリアクセス（ＤＭＡ）ストリームのうちの少なくとも１つにマッピングするステップのうちの少なくとも１つを含む、請求項１２に記載のコンピュータシステム。
【請求項１４】
前記マッピングするステップの前に、前記カーネルのうちの１つまたは複数のコードを最適化するステップをさらに含む、請求項１２に記載のコンピュータシステム。
【請求項１５】
前記データを生成する前記ステップは、
各ＤＰＥが前記カーネルのうちの１つまたは複数を内部に実装するためのコードを生成するステップと、

前記ＤＰＥアレイ内のダイレクトメモリアクセス（ＤＭＡ）およびスイッチ回路を構成するための制御コードを生成するステップと
を含む、請求項１２に記載のコンピュータシステム。
【発明の詳細な説明】
【技術分野】
【０００１】
技術分野
本開示の例は、一般に、電子回路設計に関し、特に、異種マルチコアアーキテクチャのコンパイルフローに関する。
【背景技術】
【０００２】
背景
プロセッサ、システムオンチップ（ＳｏＣ）、および特定用途向け集積回路（ＡＳＩＣ）は、デジタル信号の処理、暗号化の実行、ソフトウェアアプリケーションの実行、グラフィックスのレンダリングなどの計算作業を実行するための複数のコアを含むことができる。多くのマルチコアアーキテクチャがあるが、これらのアーキテクチャのコンパイラはいずれも、異種アーキテクチャ、特に再構成可能／プログラマブルロジックに結合されたマルチコアプロセッサ（フィールドプログラマブルゲートアレイ（ＦＰＧＡ）ファブリックなど）に直接対応していない。さらに、既存のコンパイラは、計算カーネルのプロセッサコアへのマッピング、データ構造のメモリバンクへのマッピング、ならびに、プロセッサコア間、および、プロセッサコアとプログラマブルロジックとの間のストリームデータおよびダイレクトメモリアクセス（ＤＭＡ）データのルーティングを解決しない。
【発明の概要】
【課題を解決するための手段】
【０００３】
概要
異種マルチコアアーキテクチャのコンパイルフローに関連する技法について説明する。一例では、データ処理エンジン（ＤＰＥ）アレイを有するシステムオンチップ（ＳＯＣ）のアプリケーションを実装する方法は、アプリケーションのグラフ表現を決定することであって、グラフ表現は、アプリケーションのカーネルを表すノード、および、カーネル間の通信を表すエッジを含む、グラフ表現を決定することと、グラフに基づいて、カーネルをＤＰＥアレイのＤＰＥにマッピングし、カーネルのデータ構造をＤＰＥアレイ内のメモリにマッピングすることと、ＤＰＥとＳＯＣのプログラマブルロジック内に構成されたアプリケーションの回路との間の通信チャネルをルーティングすることと、マッピングおよびルーティングの結果に基づいてアプリケーションを実装するためにＳＯＣをプログラミングするための実装データを生成することとを含む。
【０００４】
別の例では、プロセッサによって実行されると、データ処理エンジン（ＤＰＥ）アレイを有するシステムオンチップ（ＳＯＣ）のアプリケーションを実装する方法をプロセッサに実行させる命令を格納されている非一時的コンピュータ可読媒体であって、方法は、アプリケーションのグラフ表現を決定することであって、グラフ表現は、アプリケーションのカーネルを表すノード、および、カーネル間の通信を表すエッジを含む、グラフ表現を決定することと、グラフに基づいて、カーネルをＤＰＥアレイのＤＰＥにマッピングし、カーネルのデータ構造をＤＰＥアレイ内のメモリにマッピングすることと、ＤＰＥとＳＯＣのプログラマブルロジック内に構成されたアプリケーションの回路との間の通信チャネルをルーティングすることと、マッピングおよびルーティングの結果に基づいてアプリケーションを実装するためにＳＯＣをプログラミングするための実装データを生成することとを含む。
【０００５】
別の例では、コンピュータシステムは、プログラムコードを格納するように構成されたメ

モリと、プログラムコードを実行するように構成されたプロセッサであって、プログラムコードは、アプリケーションのグラフ表現を決定することであって、グラフ表現は、アプリケーションのカーネルを表すノード、および、カーネル間の通信を表すエッジを含む、グラフ表現を決定することと、グラフに基づいて、カーネルをＤＰＥアレイのＤＰＥにマッピングし、カーネルのデータ構造をＤＰＥアレイ内のメモリにマッピングすることと、ＤＰＥとＳＯＣのプログラマブルロジック内に構成されたアプリケーションの回路との間の通信チャネルをルーティングすることと、マッピングおよびルーティングの結果に基づいてアプリケーションを実装するためにＳＯＣをプログラミングするための実装データを生成することとによって、データ処理エンジン（ＤＰＥ）アレイを有するシステムオンチップ（ＳＯＣ）のアプリケーションを実装するためのものである、プロセッサとを備える。

【０００６】

これらおよび他の態様は、以下の詳細な説明を参照して理解することができる。

図面の簡単な説明

上に列挙された特徴を詳細に理解することができるように、上で簡単に要約されたより詳細な説明は、例示的な実施態様を参照することによって得ることができ、そのいくつかは添付の図面に示されている。ただし、添付の図面は典型的な実施態様例のみを示しており、そのため、その範囲を限定するものとは考えられないことに留意されたい。

【図面の簡単な説明】

【０００７】

【図１】一例によるシステムオンチップ（ＳｏＣ）のブロック図である。

【図２】一例によるタイル回路を示すブロック図である。

【図３】一例による図２のタイル回路をより詳細に示すブロック図である。

【図４】一例による相互接続回路を示すブロック図である。

【図５】一例による回路設計システムの一例を示すブロック図である。

【図６】一例によるターゲットプラットフォームのアプリケーションの実装を示すブロック図である。

【図７】一例によるアプリケーションを示すブロック図である。

【図８】一例によるデータ処理エンジン（ＤＰＥ）アレイコンパイラを示すブロック図である。

【図９Ａ】一例による、ＤＰＥアレイを標的とするアプリケーションの一部分の例示的な有向グラフ表現を示すブロック図である。

【図９Ｂ】一例による、ＤＰＥアレイを標的とするアプリケーションの一部分の例示的な有向グラフ表現を示すブロック図である。

【図９Ｃ】一例による、ＤＰＥアレイを標的とするアプリケーションの一部分の例示的な有向グラフ表現を示すブロック図である。

【図９Ｄ】一例による、ＤＰＥアレイを標的とするアプリケーションの一部分の例示的な有向グラフ表現を示すブロック図である。

【図１０】一例による、ＤＰＥアレイ内の有向グラフの配置およびルーティングを示すブロック図である。

【図１１】一例による、ＳＯＣのデータ処理エンジンアレイのコードおよび構成データを生成する方法を示す流れ図である。

【図１２】一例による、データ処理エンジン間でアプリケーション内のカーネルをパーティショニングする方法を示す流れ図である。

【図１３】一例による、カーネルをパーティションに割り当てる方法を示す流れ図である。

【図１４】一例による、カーネルおよびカーネルクラスタをＤＰＥにマッピングする方法を示す流れ図である。

【図１５】異種マルチコアアーキテクチャのアプリケーションを実装する際のＦＩＦＯ挿入の方法を示す流れ図である。

【図１６】一例による処理システムを示すブロック図である。
【図１７Ａ】一例による図１６の処理システムの実施態様を示すブロック図である。
【図１７Ｂ】別の例による図１６の処理システムの実施態様を示すブロック図である。
【図１８】一例による、ＦＩＦＯを配置する方法を示す流れ図である。
【図１９】一例による、ＤＰＥアレイにマッピングされたアプリケーションをルーティングする方法を示す流れ図である。
【図２０】図１に示されるＳｏＣの一実施態様として使用することができる、一例によるプログラム可能なＩＣを示すブロック図である。
【図２１】一例による、図２０のプログラマブルＩＣのフィールドプログラマブルゲートアレイ（ＦＰＧＡ）実施態様を示す図である。
【発明を実施するための形態】
【０００８】
理解を容易にするために、可能な場合は、図に共通する同一の要素を示すために同一の参照符号が使用されている。一例の要素は、他の例に有益に組み込まれ得ることが企図される。
【０００９】
詳細な説明
以下、図を参照して様々な特徴を説明する。図は原寸に比例して描かれている場合と描かれていない場合があり、同様の構造または機能の要素は、図全体を通して同様の参照符号によって表されていることに留意すべきである。これらの図は、機能の説明を容易にすることのみを目的としていることに留意すべきである。これらの図は、クレームされた発明の網羅的な説明として、または特許請求されている発明の範囲の限定として意図されたものではない。さらに、図示された例は、示されたすべての態様または利点を有する必要はない。特定の例に関連して説明される態様または利点は、必ずしもその例に限定されるものではなく、そのように図示されていないか、またはそのように明示的に説明されていない場合でも、他の任意の例において実施することができる。
【００１０】
本明細書において記載されている技法は、システムオンチップ（ＳＯＣ）のマルチコアアーキテクチャのアプリケーションのグラフベースのプログラム記述を取得し、アプリケーションをマルチコアアーキテクチャにコンパイルして、各コアの実行バイナリおよびプログラム可能な構成要素の構成コードを生成するプロセスを提供する。コンパイルステップは、入力グラフ記述を内部表現に変換することと、コード分析および最適化を実行することと、グループ化すべき計算カーネルを識別すること（クラスタリングなど）と、これらのグループを特定のデータ処理エンジン（コアなど）にマッピングし、カーネルによって使用されるデータ構造をローカルメモリにマッピングすることとを含む。コンパイルステップは、ストリームスイッチを介してデータ処理エンジン間で、ならびに、プログラマブルロジックへと、および、プログラマブルロジックから、ストリームおよびダイレクトメモリアクセス（ＤＭＡ）データをルーティングすることをさらに含む。コンパイルステップは、各データ処理エンジンの実行を調整するラッパーコードを生成することと、ＤＭＡおよびストリームスイッチの構成コードを生成することと、アプリケーションを制御するために処理システムによって実行するためのプログラムを生成することとをさらに含む。これらおよびさらなる態様は、図面に関して下記に説明される。
【００１１】
図１は、一例によるデータ処理エンジン（ＤＰＥ）アレイ１０５を含むデバイス１００のブロック図である。例では、デバイス１００は、システムオンチップ（ＳｏＣ）タイプのデバイスである。一般に、ＳｏＣとは、互いに相互作用することが可能な２つ以上のサブシステムを含むＩＣを指す。一例として、ＳｏＣは、プログラムコードを実行するプロセッサと、１つまたは複数の他の回路とを含み得る。他の回路は、ハードワイヤード回路、プログラム可能な回路、他のサブシステム、および／またはそれらの任意の組み合わせとして実装され得る。回路は、互いに、および／またはプロセッサと協調して動作すること

ができる。ＤＰＥアレイ１０５は、デバイス１００内の格子、クラスタ、または格子縞パターンに配置構成され得る複数のデータ処理エンジン（ＤＰＥ）１１０を含む。図１は、行および列を有する２ＤアレイにＤＰＥ１１０を配置構成することを示しているが、例は、この配置構成に限定されない。さらに、アレイ１０５は、任意のサイズであり得、ＤＰＥ１１０によって形成された任意の数の行および列を有することができる。

【００１２】

一実施形態では、ＤＰＥ１１０は同一である。すなわち、各ＤＰＥ１１０（タイルまたはブロックとも呼ばれる）は、同じハードウェア構成要素または回路を有し得る。さらに、本明細書の例は、ＤＰＥ１１０に限定されない。代わりに、デバイス１００は、任意の種類の処理要素またはデータ処理エンジンのアレイを含むことができる。さらに、ＤＰＥ１１０は、１つまたは複数の特殊なタスクを実行するための暗号化エンジンまたは他の特殊なハードウェアであり得る。したがって、ＤＰＥ１１０は、一般に、データ処理エンジンと呼ぶことができる。

【００１３】

図１では、アレイ１０５は、すべて同じタイプであるＤＰＥ１１０を含む（例えば、同種アレイ）。しかしながら、別の実施形態では、アレイ１０５は、異なるタイプのエンジンを含み得る。例えば、アレイ１０５は、ＤＰＥ１１０、暗号化エンジン、前進型誤り訂正（ＦＥＣ）エンジンなどを含み得る。アレイ１０５が同種であるかまたは異種であるかに関係なく、ＤＰＥ１１０は、下記により詳細に説明するように、ＤＰＥ１１０がメモリモジュールを共有することを可能にする、隣接ＤＰＥ１１０内のメモリモジュールへの接続を含むことができる。

【００１４】

一実施形態では、ＤＰＥ１１０は、非プログラマブルロジックから形成されている、すなわち、ハード化されている。そうすることの１つの利点は、ＤＰＥ１１０内にハードウェア要素を形成するためにプログラマブルロジックを使用することと比較して、ＤＰＥ１１０がデバイス１００内で占めるスペースをより少なくすることができることである。すなわち、ハード化または非プログラマブルロジックを使用して、プログラムメモリ、命令フェッチ／復号ユニット、固定小数点ベクトルユニット、浮動小数点ベクトルユニット、算術論理演算ユニット（ＡＬＵ）、乗加算器（ＭＡＣ）などのＤＰＥ１１０内のハードウェア要素を形成することによって、デバイス１００内のアレイ１０５のフットプリントを大幅に削減することができる。ＤＰＥ１１０はハード化され得るが、これは、ＤＰＥ１１０がプログラム可能でないことを意味するものではない。すなわち、ＤＰＥ１１０は、デバイス１００の電源がオンにされたとき、または再起動されたときに、異なる機能またはタスクを実行するように構成することができる。

【００１５】

ＤＰＥアレイ１０５はまた、ＤＰＥ１１０とデバイス１００内の他のハードウェア構成要素との間の通信インターフェースとして機能するＳＯＣインターフェースブロック１１５を含む。この例では、デバイス１００は、ＳｏＣインターフェースブロック１１５に通信可能に結合されたネットワークオンチップ（ＮｏＣ）１２０を含む。図示されていないが、ＮｏＣ１２０は、デバイス１００内の様々な構成要素が互いに通信することを可能にするために、デバイス１００全体を通じて延在することができる。例えば、物理的な実装では、ＤＰＥアレイ１０５は、デバイス１００を形成する集積回路の右上部分に配置され得る。しかしながら、それにもかかわらず、ＮｏＣ１２０を使用して、アレイ１０５は、デバイス１００全体を通じた異なるロケーションに配置され得る、例えば、プログラマブルロジック（ＰＬ）１２０、プロセッササブシステム（ＰＳ）１３０、または入力／出力（Ｉ／Ｏ）１３５などの様々なサブシステムと通信することができる。

【００１６】

ＤＰＥ１１０とＮｏＣ１２０との間のインターフェースを提供することに加えて、ＳｏＣインターフェースブロック１１５はまた、ＰＬ１２２内の通信ファブリックへの直接接続を提供することができる。一実施形態では、ＳｏＣインターフェースブロック１１５は、

ＤＰＥ１１０をＮｏＣ１２０およびデバイス１００内のアレイ１０５の近くに配置されたＰＬ１２２に通信可能に結合するための別個のハードウェア構成要素を含む。

【００１７】

図１は、ＰＬ１２２の１つのブロックを示しているが、デバイス１００は、デバイス１００内の異なるロケーションに配置することができるＰＬ１２２の複数のブロック（構成論理ブロックとも呼ばれる）を含み得る。例えば、デバイス１００は、フィールドプログラマブルゲートアレイ（ＦＰＧＡ）を形成するハードウェア要素を含み得る。しかしながら、他の実施形態では、デバイス１００は、ＰＬ１２２を一切含まなくてもよく、例えば、デバイス１００は、ＡＳＩＣである。

【００１８】

図２は、一例によるＤＰＥ１１０を示すブロック図である。ＤＰＥ１１０は、上で説明され、図１に示されているように、ＤＰＥアレイ内のＤＰＥを実装するために使用することができる。ＤＰＥ１１０は、コア２０２、メモリ２０４、ＤＰＥ相互接続２０９、およびサポート回路２１４を含む。ＤＰＥ相互接続２０９は、ストリーミング相互接続２１０およびメモリマップ（ＭＭ）相互接続２１２を含む。一例では、サポート回路２１４は、デバッグ／トレース／プロファイル回路２１６、ハードウェア（ＨＷ）同期回路（「ＨＷロック２１８」）、およびダイレクトメモリアクセス（ＤＭＡ）回路（「ＤＭＡ２２０」）を含む。メモリ２０４は、プログラムメモリ（「ＰＭ２０６」）およびデータメモリ（「ＤＭ２０８」）を含む。

【００１９】

コア２０２は、ＰＭ２０６に格納された命令（複数可）に従ってデータを処理するための１つまたは複数の計算ユニットを含む。一例では、コア２０２は、超長命令語（ＶＬＩＷ）プロセッサ、単一命令複数データ（ＳＩＭＤ）もしくはベクトルプロセッサ、またはＶＬＩＷ　ＳＩＭＤ／ベクトルプロセッサを含む。一例では、ＰＭ２０６はコア２０２にプライベートである（例えば、ＰＭ２０６は、ＤＰＥ２００内のコア２０２によって使用するためにのみ命令（複数可）を格納する）。一例では、ＰＭ２０６は、シングルポートランダムアクセスメモリ（ＲＡＭ）を含む。ＰＭ２０６は、命令の構成およびロードのために、ＭＭ相互接続２１２に結合することができる。一例では、ＰＭ２０６は、パリティ、誤り訂正符号（ＥＣＣ）保護および報告、またはパリティとＥＣＣの両方をサポートする。例えば、ＰＭ２０６は、９ビットＥＣＣをサポートすることができ、プログラム命令（例えば、１２８ビット）における１ビットエラーまたは２ビットエラーを訂正することができる。

【００２０】

コア２０２は、ストリーミング相互接続２１０に直接結合されて、入力ストリーム（複数可）を受信し、および／または出力ストリーム（複数可）を提供することができる。さらに、コア２０２は、ＤＰＥ２００内のＤＭ２０８に対してデータを読み書きすることができる。下記にさらに説明するように、ＤＰＥ２００のコア２０２は、１つまたは複数の隣接するタイル回路（例えば、北側、南側、東側、および西側の隣接するタイル回路）のＤＭにアクセスすることもできる。一例では、下記にさらに説明するように、コア２０２はまた、アキュムレータ出力を転送するための、１つまたは複数の隣接タイル内のデータ処理エンジンとの直接接続部（例えば、入力および出力カスケード接続部（複数可））を含むことができる。一例では、コア２０２は、ＤＰＥ２００内のＤＭ２０８および隣接するタイル（複数可）内の他のＤＭ（複数可）を、メモリの１つの連続したブロックとして見る。コア２０２はまた、ＨＷロック２１８へのインターフェースおよびデバッグ／トレース／プロファイル回路２１６へのインターフェースを含むことができる。デバッグ／トレース／プロファイル回路２１６は、トレース、デバッグ、および／またはプロファイル回路を含むことができる。

【００２１】

ＭＭ相互接続２１２は、構成要素間のアドレストランザクションを使用してデータを送信するように構成されたＡＸＩメモリマップ相互接続などであり得る。一例では、ＭＭ相互

接続２１２は、ＤＰＥ２００の構成、制御、およびデバッグ機能のために使用される。ＭＭ相互接続２１２は、アドレスに基づいてトランザクションをルーティングする１つまたは複数のスイッチを含む。回路は、ＭＭ相互接続２１２を使用して、メモリ２０４、コア２０２、ＤＭＡ２２０、およびＤＰＥ２００内の構成レジスタにアクセスすることができる。

【００２２】

ストリーミング相互接続２１０は、構成要素間でストリーミングデータを送信するように構成された高度拡張可能相互接続（Ａｄｖａｎｃｅｄ　ｅＸｔｅｎｓｉｂｌｅ　ｉｎｔｅｒｃｏｎｎｅｃｔ：ＡＸＩ）ストリーミング相互接続などであり得る。ストリーミング相互接続２１０は、ＤＰＥ２００と外部回路との間でデータを転送するために使用される。ストリーミング相互接続２１０は、データおよび制御の両方のための回路交換メカニズムおよびパケット交換メカニズムの両方をサポートすることができる。

【００２３】

一例では、下記にさらに説明するように、ＤＭ２０８は、１つまたは複数のメモリバンク（例えば、ランダムアクセスメモリ（ＲＡＭ）バンク）を含み得る。ＤＭＡ２２０は、ストリーミング相互接続２１０とＤＭ２０８との間に結合される。ＤＭＡ２２０は、データをストリーミング相互接続２１０からＤＭ２０８に移動させ、データをＤＭ２０８からストリーミング相互接続２１０に移動させるように構成される。このようにして、外部回路（例えば、プログラマブル論理内に構成された回路またはＩＣの埋め込み処理システム内の回路）は、ＤＭＡを使用してストリーミング相互接続２１０を介してＤＭ２０８に対してデータを読み書きすることができる。ＤＭＡ２２０は、ＭＭ相互接続２１２および／またはストリーミング相互接続２１０を介して制御され得る。一例では、ＤＭ２０８は、パリティ、誤り訂正符号（ＥＣＣ）保護および報告、またはパリティとＥＣＣの両方をサポートする。例えば、ＤＭ２０８は、９ビットＥＣＣ（例えば、１２８ビットデータ）をサポートすることができる。

【００２４】

ＨＷロック２１８は、コア２０２、別のタイル内の別のデータ処理エンジン、またはＤＭＡ２２０によるアクセスのためにＤＭ２０８の特定のメモリバンクをロックするために使用されてもよい。ＨＷロック２１８は、隣接し合うタイル内の隣接し合うデータ処理エンジン間、コア２０２とＤＭＡ２２０との間、およびコア２０２と外部回路（例えば、外部プロセッサ）との間に同期をもたらす。ＨＷロック２１８はまた、１つまたは複数のメモリバンクに格納され得るかまたは単一のメモリバンクの一部分に格納され得る、ＤＭ２０８内の特定のバッファをロックするために使用することもできる。デバッグ／トレース／プロファイル回路２１６は、デバッグ機能、トレース機能およびプロファイル機能を提供するように構成される。デバッグ／トレース／プロファイル回路２１６は、ＤＰＥ２００内の回路によって生成されるイベントをトレースすることができる。デバッグ／トレース／プロファイル回路２１６は、プロファイル機能、例えば、構成可能な実施カウンタを提供することができる。

【００２５】

図３は、一例によるＤＰＥ２００をより詳細に示すブロック図である。この例では、ＤＰＥ２００は、コア２０２、メモリモジュール３５１、およびＤＰＥ相互接続２０９を含む。コア２０２は、計算回路２０３およびＰＭ２０６を含む。メモリモジュール３５１は、メモリインターフェース３０２Ｎ、３０２Ｓ、３０２Ｅおよび３０２Ｗ（総称してメモリインターフェースまたは個々に「メモリＩＦ」）、ＲＡＭバンク３１８、ＨＷロック２１８、レジスタ（「レジスタ３０４」）、ＤＭＡインターフェース２０４Ａ、およびＤＭＡインターフェース２２０Ｂを含む。計算回路２０３は、レジスタ（「レジスタ３０６」）を含む。ＤＰＥ相互接続２０９は、ＭＭ相互接続２１２およびストリーミング相互接続２１０（図２に示す）を含む。ＭＭ相互接続２１２およびストリーミング相互接続２１０はともに、ＲＡＭバンク３１８にアクセスすることができる。ＲＡＭバンク３１８は、バンクごとにアービトレーション論理３１９を含む。アービトレーション論理３１９は、どの

インターフェース（Ｎ，Ｓ，Ｅ，Ｗ，ＤＭＡ，外部ＰＳなど）がどのバンクにアクセスするかを制御するように構成される。ＤＰＥ相互接続２０９のさらなる詳細は、図４の例に関して下記に議論される。

【００２６】

ＤＰＥ相互接続２０９は、西側タイルへのストリーミング接続３１４Ｗと、東側タイルへのストリーミング接続３１４Ｅと、北側タイルへのストリーミング接続３１４Ｎと、南側タイルへのストリーミング接続３１４Ｓとを含む。各ストリーミング接続３１４は、１つまたは複数の独立したストリーミングインターフェース（例えばバス）を含んでおり、これらストリーミングインターフェースの各々は特定のビット幅を有する。ＤＰＥ相互接続２０９はまた、南側タイルからのメモリマップ接続３１２Ｓと、北側タイルへのメモリマップ接続３１２Ｎとを含む。北側および南側のＭＭ接続のみが図示されているが、ＤＰＥ相互接続２０９が、ＭＭ相互接続のための他の構成（例えば、東側から西側、西側から東側、北側から南側など）を含み得ることは理解されたい。ＤＰＥ相互接続２０９が、図３の例に示される以外のストリーミング接続およびメモリマップ接続の他の配置構成を含み得ることは理解されたい。概して、ＤＰＥ相互接続２０９は、少なくとも１つのストリーミング接続３１４と、少なくとも１つのメモリマップ接続３１２とを含む。

【００２７】

計算回路２０３は、西側タイル内のメモリ回路への接続３０８Ｗと、南側タイル内のメモリ回路への接続３０８Ｓと、北側タイル内のメモリ回路への接続３０８Ｎと、メモリモジュール３５１への接続３０８Ｅとを含む。計算回路２０３は、ＤＰＥ相互接続２０９へのストリーミングインターフェースを含む。計算回路２０３はまた、西側タイル内のコアからの接続３１０Ａと、東側タイル内のコアへの接続３１０Ｂ（例えば、カスケード接続）とを含む。ＤＰＥが、図３の例に示されるメモリおよびカスケード接続の他の配置構成を含み得ることは理解されたい。概して、計算回路２０３は、少なくとも１つのメモリ接続を含み、少なくとも１つのカスケード接続を含み得る。

【００２８】

メモリＩＦ３０２Ｗは、計算回路２０３のメモリ接続３０８Ｅに結合される。メモリＩＦ３０２Ｎは、北側タイル内のデータ処理エンジンのメモリ接続に結合される。メモリＩＦ３０２Ｅは、東側タイル内のデータ処理エンジンのメモリ接続に結合される。メモリＩＦ３０２Ｓは、南側タイル内のデータ処理エンジンのメモリ接続に結合される。メモリＩＦ３０２Ｗ、３０２Ｎ、３０２Ｅ、および３０２ＳはＲＡＭバンク３１８に結合される。ＤＭＡ２２０Ａは、相互接続ストリームへのメモリを処理するためにＤＰＥ相互接続２０９に結合された出力を含む。ＤＭＡ２２０Ｂは、メモリストリームへの相互接続を処理するためにＤＰＥ相互接続２０９に結合された入力を含む。レジスタ３０４およびレジスタ３０６は、ＤＰＥ相互接続２０９に結合されて、（例えば、メモリマップ相互接続を使用して）そこから構成データを受信する。

【００２９】

図４は、一例によるＤＰＥ相互接続２０９を示すブロック図である。ＤＰＥ相互接続２０９は、ストリームスイッチ４０２およびＭＭスイッチ４０４を含む。ストリームスイッチ４０２は、西側ストリームインターフェース４０６Ｗ、北側ストリームインターフェース４０６Ｎ、東側ストリームインターフェース４０６Ｅ、および南側ストリームインターフェース４０６Ｓに結合される。西側ストリームインターフェース４０６Ｗは、ストリームを受信して西側タイルのＤＰＥ相互接続に供給する。北側ストリームインターフェース４０６Ｎは、ストリームを受信して北側タイルのＤＰＥ相互接続に供給する。西側ストリームインターフェース４０６Ｗは、ストリームを受信して西側タイルのＤＰＥ相互接続に供給する。南側ストリームインターフェース４０６Ｓは、ストリームを受信して南側タイルのＤＰＥ相互接続に供給する。ＭＭスイッチ４０４は、北側ＭＭインターフェース４０８Ｎおよび南側ＭＭインターフェース４０８Ｓに結合される。北側ＭＭインターフェース４０８Ｎは、北側タイル内のＤＰＥ相互接続に結合される。南側ＭＭインターフェース４０８Ｓは、南側タイル内のＤＰＥ相互接続に結合される。

【００３０】
ストリームスイッチ４０２は、先入れ先出し（ＦＩＦＯ）回路（ＦＩＦＯ４１２）およびレジスタ（レジスタ４１０）を含む。ＦＩＦＯ４１２は、ストリームスイッチ４０２を通過するストリームをバッファするように構成される。レジスタ４１０は、ストリームスイッチを通るストリームのルーティングを制御するストリームスイッチ４０２のための構成データを格納する。レジスタ４１０は、ＭＭスイッチ４０４から構成データを受信し得る。ストリームスイッチ４０２は、計算回路２０３への付加的インターフェースと、ＤＭＡ回路２２０への付加的インターフェースとを含み得る。ストリームスイッチ４０２は、制御ストリームを送受信し、（例えば、デバッグ／トレース／プロファイル回路２１６から）トレースストリームを受信することができる。
【００３１】
図５は、一例による回路設計システム５００の一例を示すブロック図である。回路設計システム５００は、入出力（ＩＯ）デバイス５１２、ディスプレイ５１４、およびＳＯＣ１００に結合されたコンピュータ５０１を含む。コンピュータ５０１は、中央処理装置（ＣＰＵ）５０２、システムメモリ５０８、様々なサポート回路５０４、ストレージ５２０、およびＩＯインターフェース５０６などのコンピューティングデバイスの従来の構成要素を含むことができるハードウェアプラットフォーム５１８を含む。ＣＰＵ５０２は、１つまたは複数のマイクロプロセッサを含むことができる。ＣＰＵ５０２は、本明細書で説明される１つまたは複数の動作を実行する命令を実行するように構成される。命令は、システムメモリ５０８、ストレージ５２０、またはハードウェアプラットフォーム５１８内の他の任意のメモリ（例えば、キャッシュメモリ）に格納することができる。システムメモリ５０８は、情報を記憶するデバイスを含み、例えば、ランダムアクセスメモリ（ＲＡＭ）、読み出し専用メモリ（ＲＯＭ）、またはそれらの組み合わせを含むことができる。ストレージ５２０は、ハードディスク、フラッシュメモリモジュール、ソリッドステートディスク、光ディスクなどのようなローカルストレージデバイスを含む。ストレージ５２０はまた、１つまたは複数のネットワークデータストレージシステムとの通信するように構成されたインターフェースを含むことができる。サポート回路５０４は、従来のキャッシュ、電源、クロック回路、データレジスタ、ＩＯインターフェースなどを含むことができる。ＩＯインターフェース５０６は、当該技術分野で知られているコンピュータ５０１への従来のインターフェースを含む。ＩＯインターフェース５０６は、従来のキーボード、マウスなどを含むことができるＩＯデバイス５１２に結合することができる。ＩＯインターフェース５０６はまた、ＧＵＩ５１６をユーザに提示することができるディスプレイ５１４に結合することができる。
【００３２】
コンピュータ５０１は、オペレーティングシステム（ＯＳ）５２２および設計ツール５１０を含むソフトウェアプラットフォームをさらに含む。ＯＳ５２２および設計ツール５１０は、ＣＰＵ５０２によって実行される命令を含む。ＯＳ　５２２は、Ｌｉｎｕｘ（登録商標）、Ｍｉｃｒｏｓｏｆｔ　Ｗｉｎｄｏｗｓ（登録商標）、Ｍａｃ　ＯＳ（登録商標）などの既知のオペレーティングシステムを含んでもよい。設計ツール５１０は、ハードウェアプラットフォーム５１８へのインターフェースを提供するＯＳ５２２内で実行されるアプリケーションである。設計ツール５１０の動作は、下記に説明される。本明細書に記載の技術を含むように適合させることができる例示的な設計ツールは、カリフォルニア州サンノゼ所在のＸｉｌｉｎｘ，Ｉｎｃ．から入手可能なＶｉｖａｄｏ（登録商標）設計スイートであるが、他の回路設計ツールも同様に適合させることができる。
【００３３】
異種マルチコアアーキテクチャのコンパイラフロー
図６は、一例によるターゲットプラットフォームのアプリケーションの実装を示すブロック図である。設計ツール５１０は、ソフトウェアコンパイラ（「ＳＷコンパイラ６０２」）、ＤＰＥアレイコンパイラ６０４、およびハードウェア設計ツール（「ＨＷ設計ツール６０６」）を含む。設計ツール５１０は、アプリケーション６０８、ユーザ制約６１０、

およびハードウェアベースプラットフォーム（「ＨＷベースプラットフォーム６１２」）を受信する。ユーザは、ＳｏＣ１００向けに実装されるアプリケーション６０８を定義する。アプリケーション６０８は、ＰＬ１２２、ＰＳ１３０、およびＤＰＥアレイ１０５にマッピングされる部分を含む。ＳＷコンパイラ６０２は、任意のプログラミング言語（例えば、Ｃ、Ｃ＋＋など）を使用して定義されたソースコードを含むことができる、ＰＳ１３０にマッピングされるアプリケーション６０８の部分をコンパイルするように構成される。ＨＷ設計ツール６０６は、任意のハードウェア記述言語（ＨＤＬ）、レジスタ転送言語（ＲＴＬ）、高位合成（ＨＬＳ）モデル、並列プログラミング言語（例えば、ＳｙｓｔｅｍＣ）などを使用して定義された回路記述を含むことができる、ＰＬ１２２にマッピングされるアプリケーション６０８の部分を実装するように構成される。ＤＰＥアレイコンパイラ６０４は、下記にさらに定義されるＤＰＥアレイ１０５を標的とするアプリケーション６０８の部分をコンパイルするように構成される。

【００３４】

図７は、一例によるアプリケーション６０８を示すブロック図である。ユーザは、設計ツール５１０と相互作用して、ＳＯＣ１００を標的とするアプリケーション６０８を定義する。この例では、アプリケーション６０８は、入力回路７０２、カーネル７０４、カーネル７０６、出力回路７０８、および制御ソフトウェア７１０を含む。入力回路７０２は、カーネル７０４に結合され、カーネル７０４にデータを提供する。カーネル７０４は、カーネル７０６に結合され、カーネル７０６にデータを提供する。カーネル７０６は、出力回路７０８に結合され、出力回路７０８にデータを提供する。制御ソフトウェア７１０は、カーネル７０４およびカーネル７０６を制御し、それらにデータを提供する。一般に、アプリケーション６０８は、設計に応じて特定の方法で互いに接続された１つまたは複数の回路、１つまたは複数のカーネル、および制御ソフトウェアを含む。

【００３５】

この例では、入力回路７０２は、外部システム／回路と通信し、かつ処理のためにカーネル７０４にデータを提供する構成されたデジタルロジック（および任意選択的にアナログロジック）を含む。入力回路７０２は、ＰＬ１２２にマッピングされる。同様に、出力回路７０８は、外部システム／回路と通信し、かつ処理されたカーネル７０６からのデータを受信するように構成されたデジタルロジック（および任意選択的にアナログロジック）を含む。出力回路７０８は、ＰＬ１２２にマッピングされる。一例では、カーネル７０４および７０６は、データプロセッサのプログラム記述を含む。カーネル７０４および７０６は、ＤＰＥアレイ１０５にマッピングされる。制御ソフトウェア７１０は、カーネル７０４および７０６のためのコントローラのプログラム記述である。一例では、制御ソフトウェア７１０は、ＰＳ１３０にマッピングされる。

【００３６】

図６に戻ると、実装データ６１４は、ストリームスイッチ構成コード６１６、ＤＭＡエンジン構成コード６１８、ＰＳバイナリ６２４、ＤＰＥバイナリ６２０、およびＰＬビットストリーム６２２を含む。ＳＷコンパイラ６０２は、ＰＳ１３０（例えば、制御ソフトウェア７１０）を標的とするアプリケーション６０８のソースコードからＰＳバイナリ６２４を生成する。ＰＳバイナリ６２４は、特定のマイクロプロセッサアーキテクチャ（例えば、ｘ８６、ＡＲＭ（登録商標）など）を標的とするように構成される。ＨＷ設計ツール６０６は、ＰＬ１２２（例えば、入力回路７０２および出力回路７０８）を標的とするアプリケーション６０８の部分からＰＬビットストリーム６２２を生成する。ＰＬビットストリーム６２２は、特定のＳＯＣデバイスを標的とする。ＤＰＥアレイコンパイラ６０４は、ＤＰＥアレイ１０５（例えば、カーネル７０４および７０６）を標的とするアプリケーション６０８の部分に基づいて、ストリームスイッチ構成コード６１６、ＤＭＡエンジン構成コード６１８、およびＤＰＥバイナリ６２０を生成する。ストリームスイッチ構成コード６１６は、ＤＰＥ相互接続２０９においてストリームスイッチ４０２をプログラミングするためのデータを含む。ＤＭＡエンジン構成コード６１８は、ＤＰＥ１１０のメモリモジュール３５１においてＤＭＡ回路２２０をプログラミングするためのデータを含む

。ＤＰＥバイナリ６２０は、ＤＰＥ１１０のコア２０２内の計算回路２０３によって実行するためのコードを含む。

【００３７】

実装出力６１４は、ターゲットプラットフォーム６２６上での実装のために構成されている。ターゲットプラットフォーム６２６は、シミュレーションプラットフォーム（「シミュレーション６２８」）、エミュレーションプラットフォーム（「エミュレーション６３０」）、およびハードウェアプラットフォーム（「ハードウェア６３２」）を含む。ハードウェア６３２は、ＳＯＣ１００を含む。シミュレーションプラットフォーム６２８およびエミュレーションプラットフォーム６３０は、ハードウェア６３２をシミュレート／エミュレートする。

【００３８】

図８は、一例によるＤＰＥアレイコンパイラ６０４を示すブロック図である。ＤＰＥアレイコンパイラ６０４は、フロントエンド８０６、マッパ８０８、バックエンド８１０、およびシングルコアコンパイラ８１２を含む。ＤＰＥアレイコンパイラ６０４は、個別の構成要素を有するものとして説明されているが、これらの構成要素の機能は、図８に示される例とは異なる構造においてより多いまたは少ない構成要素を使用して実装することができることを理解されたい。ＤＰＥアレイ１０５を標的とするアプリケーション６０８の部分は、ユーザ定義のグラフ記述８０２およびカーネルソースコード８０４を含む。ユーザ定義のグラフ記述８０２は、構成要素およびそれらの接続性（例えば、入力回路７０２、カーネル７０４および７０６、出力回路７０８、ならびに制御ソフトウェア７１０）を記述する。カーネルソースコード８０４は、ＤＰＥアレイ１０５のＤＰＥ１１０において実装される機能のプログラム記述を提供する。

【００３９】

ユーザ定義のグラフ記述８０２は、様々なプログラミング言語（例えば、Ｃ、Ｃ＋＋など）またはデータ構造言語（例えば、ＸＭＬ、ＪＳＯＮなど）を使用して指定することができる。Ｃ＋＋で指定されたユーザ定義のグラフ記述８０２の一例を下記に示す。

【００４０】

【表１】

```
using namespace cardano;
class radio :
        cardano::graph {
public:
        cardano::kernel a,b,c,d,e,f;
radio() {
        a = kernel::create(polarclip);
        b = kernel::create(feedback);
        c = kernel::create(equalizer);
        d = kernel::create(fir_tap11);
        e = kernel::create(fir_tap7);
        f = kernel::create(scale);
        fabric<fpga>(a);
         fabric<fpga>(f);
         connect<stream, window<64,8> >      ( a.out[0], b.in[0] );
         connect<window<32> >                ( b.out[0], c.in[0] );
         connect<window<32, 24> >            ( c.out[0], d.in[0] );
         connect<window<32, 16> >            ( d.out[1], e.in[0] );
         connect<window<32, 8> >             ( e.out[0], b.in[1] );
         connect<window<16>, stream >        ( d.out[0], f.in[0] );
        }
  }
```

【００４１】

上記の例では、無線クラスはグラフ作成プリミティブを有するクラスライブラリ（ｃａｒｄａｎｏ）から導出される。これらのプリミティブを使用して、ユーザ定義のグラフ記述８０２は、計算ノードａ、ｂ、ｃ、ｄ、ｅ、およびｆを定義する。計算ノードａおよびｆは、ＰＬ１２２にマッピングされた回路である。計算ノードｂ、ｃ、ｄ、およびｅは、ＤＰＥアレイ１０５にマッピングされたカーネルである。回路ａは、ＤＭＡストリーミング接続を使用してカーネルｂに接続される。カーネルｂはカーネルｃに接続されており、カーネルｃはカーネルｄに接続されており、カーネルｄはカーネルｅに接続されており、カーネルｅはカーネルｂに接続されており、そのような各接続はＤＰＥ１０５内のメモリブロックを介して行われる。カーネルｄは、ＤＭＡストリーミング接続を介して回路Ｆに接続されている。

【００４２】
ユーザ定義のグラフ記述８０２はまた、プラットフォームのトップレベルの記述を含むことができる。例えば、記述は以下のとおりである。
【００４３】
【表２】

```
radio mygraph;
simulation::platform<1,1> platform("in.txt", "out.txt");
connect<> net0(platform.src[0], mygraph.in);
connect<> net1(platform.sink[0], mygraph.out);
```

【００４４】
上記の例では、ユーザは無線クラス（ｍｙｇｒａｐｈ）をインスタンス化し、シミュレーションターゲットを設定する。ユーザは、ＰＳ１３０上で実行される制御プログラムを次のように指定することもできる。
【００４５】
【表３】

```
int main(void){
        mygraph.init();
        mygraph.run();
        mygraph.end();
        return 0;
}
```

【００４６】
上記の例では、ユーザはシミュレーションプラットフォームでのシミュレーションのために、ｍｙｇｒａｐｈを初期化し、ｍｙｇｒａｐｈを実行し、ｍｙｇｒａｐｈを終了する。
【００４７】
カーネルソースコード８０４は、ＤＰＥ１１０を標的とする各計算ノード（例えば、上記の例の計算ノードｂ、ｃ、ｄ、およびｅ）のソースコード記述を提供する。カーネルソースコード８０４は、任意のプログラミング言語（例えば、Ｃ、Ｃ＋＋など）を使用して定義することができる。計算ノードｃ（イコライザ）を定義するためのＣ＋＋ソースコードの例を下記に示す。
【００４８】

【表４】

```
void equalizer (input_window_cint16 * inputw, output_window_cint16 * outputw) {
  ...
  v32cint16 sbuff   = null_v32cint16();
  for ( unsigned i=0; i<LSIZE; i++)
    chess_loop_range(2,)
    chess_prepare_for_pipelining
  {
    v4cacc48 acc;
    v8cint16 coe, vdata;
    coe = *coeff++;    // LD coefficients 0:7
    window_readincr(inputw, vdata);
    sbuff = upd_w0(sbuff, vdata); // LDw0 data 0:7    - 0:7|X|X|X
    acc = mul4_nc(     sbuff,   1, 0x3210, 1, coe, 0, 0x0000, 1); // d01:d05 | c0:c1
    acc = mac4_nc(acc, sbuff,   3, 0x3210, 1, coe, 2, 0x0000, 1); // d03:d07 |
c2:c3
    window_readincr(inputw, vdata);
    sbuff = upd_w1(sbuff, vdata);
    acc = mac4_nc(acc, sbuff,   5, 0x3210, 1, coe, 4, 0x0000, 1); // d05:d09 |
c4:c5
    acc = mac4_nc(acc, sbuff,   7, 0x3210, 1, coe, 6, 0x0000, 1); // d07:d11 |
c6:c7
    coe = *coeff++;    // LD coefficients 8:15
    window_readincr(inputw, vdata);
    sbuff = upd_w2(sbuff, vdata); // LDw2 data 16:23       - 0:7|8:15|16:23|X
    ...
    window_writeincr(outputw,fsrs(acc,shift));
    ...
  }
}
```

【００４９】

この例では、計算ノードｃ（イコライザ）は、計算ノードの入力を定義する入力パラメータを有するＣ／Ｃ＋＋関数を使用して実装されている。コードは、実施のスケジューリングを支援するプラグマを含む。コードは、ベクトル化された計算を実行するための組み込み関数と、データにアクセスするためのアプリケーションプログラミングインターフェース（ＡＰＩ）とを含む。

【００５０】

フロントエンド８０６は、ユーザ定義のグラフ記述８０２を処理し、内部表現として有向グラフを生成するように構成される。有向グラフでは、ノードは計算ノードを表し、エッ

ジは計算ノード間の接続を表す。マッパ８０８は、デバイス記述８１４に基づいて、ターゲットデバイス内のＤＰＥアレイ１０５の有向グラフを実装するように構成される。マッパ８０８は、計算ノードをグループにパーティショニングし、パーティショニングされた計算ノードをＤＰＥ１１０にマッピングする。バックエンド８１０は、ＤＰＥ１１０とＰＬ１２２内の回路との間の接続をルーティングし、次にシングルコアコンパイラ８１２を呼び出してＤＰＥバイナリを生成し、また、マッパ８０８の出力に基づいてストリームスイッチおよびＤＭＡ構成コードを生成するように構成される。

【００５１】

図９Ａ～図９Ｄは、一例による、ＤＰＥアレイ１０５を標的とするアプリケーション６０８の一部分の例示的な有向グラフ表現を示すブロック図である。図１０は、ＤＰＥアレイ１０５内の有向グラフの配置およびルーティングを示すブロック図である。図９Ａ～図９Ｄの有向グラフは、計算ノードａ、ｂ、ｃ、ｄ、ｅ、およびｆを有するユーザ定義のグラフ記述８０２について上で説明した例を表し、ここで、計算ノードａおよびｆは、ＰＬ１２２にマッピングされ、計算ノードｂ、ｃ、ｄ、およびｅは、ＤＰＥアレイ１０５にマッピングされる。図９Ａに示されるように、有向グラフは、ノードＡ　９０２、Ｂ　９０４、Ｃ　９０６、Ｄ　９０８、Ｅ　９１０、およびＦ　９１２を含む。ノードＡはノードＢに接続されており、ノードＢはノードＣに接続されており、ノードＣはノードＤに接続されており、ノードＤはノードＦおよびノードＥに接続されている。図９Ｂ～図９Ｄは、下記にさらに説明するように、マッパ８０８が様々なステップを通じて有向グラフを処理する方法を示している。

【００５２】

図１１は、一例による、ＳＯＣ１００のＤＰＥアレイ１０５のコードおよび構成を生成する方法１１００を示す流れ図である。方法１１００は、コンピュータシステム５０１上で実行される設計ツール５１０によって実行される。上記のように、設計ツール５１０は、アプリケーション６０８を処理して、ＳＯＣ１００のコードおよび構成データを生成するように構成される。方法１１００は、ＤＰＥアレイコンパイラ６０４によって実行される。

【００５３】

方法１１００は、ステップ１１０２において始まり、ここで、ＤＰＥアレイコンパイラ６０４は、アプリケーション６０８を解析し、設計のユーザ定義のグラフ記述に基づいて有向グラフを生成する。一例では、ＤＰＥアレイコンパイラ６０４は、計算ノード９０２～９１２およびそれらの間の接続を識別する。次に、ＤＰＥアレイコンパイラ６０４は、図９Ａに示されるように、ノードがカーネルであり、エッジが接続であるグラフを形成する。

【００５４】

ステップ１１０４において、ＤＰＥアレイコンパイラ６０４は、有向グラフを処理して、データ処理エンジンのコア上で実行することができるグループにカーネルをクラスタ化する。この例では、カーネルは、計算ノードＢ　９０４、Ｃ　９０６、Ｄ　９０８、およびＥ　９１０から選択される。すなわち、各ＤＰＥ１１０は、１つまたは複数のカーネルを実行することができ、ＤＰＥアレイコンパイラ６０４は、どのカーネルを組み合わせてグループとして実行することができるかを決定する。図９Ｂに示すように、ＤＰＥアレイコンパイラ６０４は、ノードＢ９０４およびノードＣ９０６を組み合わせてグループ９０５にし、ノードＤ９０８およびノードＥ９１０を組み合わせてグループ９０７にすることができる。

【００５５】

ステップ１１０６において、ＤＰＥアレイコンパイラ６０４は、パフォーマンスを改善し、カーネルクラスタのメモリフットプリントを削減するために、最適化のために各カーネルを定義するコードを処理する。ステップ１１０８において、ＤＰＥアレイコンパイラ６０４は、カーネル（または存在する場合はカーネルクラスタ）をＤＰＥアレイ１０５内のＤＰＥ１１０にマッピングする。図１０に示されるように、ノードＢ９０４およびＣ９０

６によって表されるカーネルは、１つのＤＰＥのコア２０２－１にマッピングされ、ノードＤ９０８およびＥ９１０によって表されるカーネルは、別のＤＰＥのコア２０２－３にマッピングされる。

【００５６】

ステップ１１１０において、ＤＰＥアレイコンパイラ６０４は、カーネルコードにおいて定義されているデータ構造を、対応するＤＰＥ１１０内のローカルメモリバンクに、または非ローカル通信のためのＤＭＡにマッピングする。上記のように、互いに近接しているＤＰＥはメモリバンクを共有することができる。共有メモリを介した通信は、本明細書でさらに説明するように、単一バッファまたは二重バッファにすることができる。ただし、場合によっては、ＤＰＥが別のＤＰＥから十分に離れており、結果、ＤＰＥ上で実行されているカーネルがＤＭＡ通信を必要とすることがある。図９Ｃに示されるように、ノードＡ９０２とＢ９０４との間の通信は、二重バッファ９１４を介して行われる。ノードＢ９０４とノードＣ９０６との間の通信は、単一バッファ９１６を介して行われる。ノードＣ９０６とノードＤ９０８との間の通信は、二重バッファ９１８を介して行われる。ノードＤ９０８とノードＥ９１０との間の通信は、単一バッファ９２０を介して行われる。ノードＥ９１０とノードＢ９０４との間の通信は、二重バッファ９２２を介して行われる。そして、ノードＤ９０８とノードＦ９１２との間の通信は、二重バッファ９２４を介して行われる。図１０に示されるように、バッファ９１４は、メモリモジュール３５１－１内のＲＡＭバンク３１８にマッピングされる。バッファ９１８は、メモリモジュール３５１－２内のＲＡＭバンク３１８にマッピングされる。バッファ９２４は、メモリモジュール３５１－４のＲＡＭバンク３１８にマッピングされる。メモリバッファ９１６、９２０、および９２２は、メモリモジュール３５１－３のＲＡＭバンク３１８にマッピングされる。

【００５７】

ステップ１１１２において、ＤＰＥアレイコンパイラ６０４は、ＤＰＥアレイ１０５とＰＬ１２２との間で通信チャネルを配分することができる。これは例えば、ＰＬ１２２（例えば、計算ノードａおよびｆ）内で構成された入出力回路への通信チャネルである。例えば、図１０に示されるように、ノードＡ９０２は、メモリモジュール３５１－１内のＤＭＡ２２０とのＤＭＡ通信向けに構成されている。ノードＦ９１２は、メモリモジュール３５１－４内のＤＭＡ２２０とのＤＭＡ通信向けに構成されている。

【００５８】

ステップ１１１４において、ＤＰＥアレイコンパイラ６０４は、カーネル間でコア対コアおよびコア対ＤＭＡ（必要な場合）同期のためにロックを配分する。図９Ｄに示されるように、ＤＰＥアレイコンパイラ６０４は、ノードＡ９０２とＢ９０４との間でロック９２４を配分し、ノードＣ９０６とＤ９０８との間でロック９２８を配分し、ノードＥ９１０とＢ９０４との間でロック９２６を配分し、ノードＤ９０８とＦ９１２との間でロック９３０を配分する。ロック９２４、９２６、９２８、および９３０は、ノード間の二重バッファの同期を提供する。

【００５９】

ステップ１１１６において、ＤＰＥアレイコンパイラ６０４は、コア間で、およびストリームスイッチ（例えば、回線および／またはパケット交換）を介してＰＬ１２２から／へのストリームデータ通信をルーティングする。ステップ１１１８において、ＤＰＥアレイコンパイラ６０４は、各ＤＰＥ１１０のラッパーコードを生成し、ラッパーコードをコンパイルして、ＤＰＥバイナリを生成する。ステップ１１２０において、ＤＰＥアレイコンパイラ６０４は、ＤＭＡストリームスイッチを構成するための制御コードを生成する。ステップ１１２２において、ＤＰＥアレイコンパイラ６０４は、データ処理エンジンアレイ内で実行されているカーネルにアクセスして制御するために処理システム上で実行されている制御ソフトウェアのためのアプリケーションプログラミングインターフェース（ＡＰＩ）を生成する。

【００６０】

ヒューリスティックパーティショナ

上記の方法１１００において、ＤＰＥアレイコンパイラ６０４は、カーネルを、ＤＰＥ１０上で実行できるグループにクラスタ化する（ステップ１１０４）。計算上、パーティショニング問題は非多項式（ＮＰ）ハードである。これは、競合を伴うビンパッキングの古典的な問題がパーティショニング問題に還元されることに起因する。本明細書における例では、扱いやすさのために、貪欲なヒューリスティックがパーティショニングアルゴリズムにおいて使用される。グラフベースのプログラミングモデルでは、各カーネルは関連する実行時間比を有し、これは、サイクルバジェットと比較した実行時間の上限を示す。ともにクラスタ化されたカーネルの実行時間比の合計は、１を超えてはならない。さらに、ユーザは、カーネル間のコロケーション制約、またはカーネルの絶対ロケーション制約を指定することもできる。これは、パーティションへのカーネルの配分、および、データ処理エンジンへのカーネル／パーティションのマッピングに影響を与える。以前の取り組みと比較して、本明細書において説明するパーティショニングスキームは、絶対制約、相対制約、および派生制約を含む様々な制約を処理するという点で独特である。また、これらの技法は、パーティションの作成中にいくつかの制約を動的に推測し、結果、実行可能なパーティションからコアへのマッピングを見つけることができる。さらに、これらの技法は、ユーザが、各々が多基準の目的関数を有する種々のパーティショニングヒューリスティックから選択することを可能にする。

【００６１】

図１２は、一例による、ＤＰＥ間でアプリケーション内のカーネルをパーティショニングする方法１２００を示す流れ図である。方法１２００は、上記のステップ１１０４の一部として実行することができる。方法１２００は、ステップ１２０２において始まり、ここで、ＤＰＥアレイコンパイラ６０４は、有向グラフ（例えば、上記のステップ１１０２で形成された）を受信する。有向グラフでは、ノードはカーネルであり、エッジはカーネル間のデータフローを表す。ユーザは、特定のカーネルを特定のデータ処理エンジンに配置する必要があるなど、絶対制約を定義することができる。ユーザはまた、特定のカーネルセットを同じデータ処理エンジンに配置する必要があるなど、相対制約を定義することもできる。ステップ１２０４において、ＤＰＥアレイコンパイラ６０４は、ユーザ定義の制約（「ユーザ制約」）に基づいて、いくつかのカーネルを「マクロノード」にグループ化する。この時点で、グラフはノードおよびマクロノードを含み、ノードは単一のカーネルを含み、マクロノードは２つ以上のカーネルを含む。以降、この説明ではノードとマクロノードとを区別せず、ノードおよびカーネルを交換可能に使用する。

【００６２】

ステップ１２０６において、ＤＰＥアレイコンパイラ６０４は、それらの実行時間比および重要度に基づいてグラフ内のノードをソートする。これは２つのステップにおいて行われる。まず、ＤＰＥアレイコンパイラ６０４は、静的レベルに基づいてノードをソートして、リストＬ１を作成する。ノードｎの静的レベルは、ｎからグラフ内の任意のシンクへの最大累積実行時間比として計算される。次に、ＤＰＥアレイコンパイラ６０４は、Ｌ１をスキャンして、未探索のノードｍを見つける。ｍをルートとして、深さ優先の探索を優先しながら、逆後行順（ＲＰＯ）トラバーサルを実行する。この説明では、このトラバーサルを深さ優先ＲＰＯと呼ぶ。ＲＰＯトラバーサルにおいて探索されたすべてのノードは、新しいリストＬ２に追加される。次に、新しい未探索ノードがＬ１から選択され、Ｌ１内のすべてのノードが探索されるまでステップ１２０６が繰り返される。ＲＰＯトラバーサルは局所性を活用し、プロデューサ－コンシューマカーネルを同じパーティションに配置する可能性を高め、クリティカルパスが悪化しない可能性を高める。

【００６３】

ステップ１２０８において、ＤＰＥアレイコンパイラ６０４は、ソートされたノードを一度に１つずつ処理し、それらを最終的なパーティションに配置する。各ノードは、既存のパーティションまたは新しいパーティションに配置することができる。割り当ては、選択したカーネルと各パーティションの既存のカーネルとの間の制約を決定することによって実行される。例えば、ＤＰＥアレイコンパイラ６０４は、各パーティションの実行時間比

を１以下に維持する（例えば、所与のデータ処理エンジンがオーバーコミットにならないようにする）（ステップ１２１０）。したがって、ｂ＿ａ内のカーネルの実行時間比とカーネルｋ＿ａの実行時間比との合計が１を超える場合、カーネルｋ＿ａをパーティションｂ＿ａに割り当てることはできない。

【００６４】

別の例では、絶対制約のあるカーネルを有するパーティションは絶対制約を継承する（ステップ１２１２）。したがって、カーネルｋ＿ａが、カーネルを特定のデータ処理エンジンに固定する絶対制約を有し、パーティショニングアルゴリズムがｋ＿ａをパーティションｂ＿ａにマッピングする場合、ｋ＿ａの絶対制約はパーティションｂ＿ａに拡張される。その後、ＤＰＥアレイコンパイラ６０４は、異なるカーネルｋ＿ｂをパーティションｂ＿ａに追加しない。ここで、ｋ＿ｂは、それをｋ＿ａとは異なるデータ処理エンジンにマッピングする絶対制約を有する。

【００６５】

別の例では、ＤＰＥアレイコンパイラ６０４は、パーティションがデータ処理エンジンアレイの物理的制約を満たすことを保証する（ステップ１２１４）。例えば、パーティションには、所与のアーキテクチャに対して定義された数を超える入出力ストリームポートを含めることはできない。同じパーティション内の２つのカーネルは、それらの間にストリーム接続、カスケード接続、非同期接続など、特定のタイプの接続を有することはできない。

【００６６】

別の例では、ＤＰＥアレイコンパイラ６０４は、パーティショニングプロセス（ステップ１２１６）から生じる任意の動的なロケーション制約をすべて満たす。パーティショニングが進むにつれて、いくつかのロケーション制約が、交差する二重バッファデータエッジを有する２つのパーティション間で動的に発生する可能性がある。このようなパーティションは、アレイ内の隣接するデータ処理エンジンにマッピングされる必要がある場合がある。

【００６７】

ステップ１２０８において、パーティショニングアルゴリズムは、すべての制約を追跡し、カーネルをパーティションに割り当てる間、それらを尊重する。制約の競合のためにカーネルを任意の既存のパーティションに追加することができない場合、ＤＰＥアレイコンパイラ６０４は新しいパーティションを作成する。ただし、カーネルを追加することができるパーティションが複数ある場合は、次の２つの可能性がある。（１）ＤＰＥアレイコンパイラ６０４は、既存のパーティションの１つにカーネルを追加することができる、または（２）ＤＰＥアレイコンパイラ６０４は、カーネルを新しいパーティションに追加することができる。第１のオプションは、開かれるパーティションの数を最小限に抑え、これは、消費電力に直接影響する。第２のオプションは、全体的な実行待ち時間を短縮するのに役立つ。ユーザは、異なるアプリケーションに対して異なる目的を有する可能性がある（例えば、電力使用量の削減対実行時間の削減）ため、ＤＰＥアレイコンパイラ６０４は、ユーザに２つの実施態様を提供することができる。（１）可能な場合はいつでも、パーティションの数を最小限に抑える、すなわち、競合のない既存のパーティションの１つにカーネルを追加する。（２）クリティカルパス長を悪化させない場合にのみカーネルを競合のないパーティションに追加する、それ以外の場合は、新しいパーティションが作成される。どちらの場合も、カーネルを複数のパーティションに追加することができるとアルゴリズムが判断した場合、パーティションにまたがる二重バッファエッジの数を最小限に抑えるパーティションが優先される。

【００６８】

ステップ１２１８において、設計ツールは、重要度に基づいて、各パーティション内のカーネルの実行順序を決定する。パーティショニング後、各パーティション内のカーネルは順番に実行される。実行待ち時間の増加を回避するために、所与のパーティション内のカーネルは、その重要度に基づいて実行される。

【００６９】
パーティショニング問題のいくつかの制約は、整数線形計画（ＩＬＰ）問題として定式化することもできる。これは、ＩＬＰソルバを使用して解決することができる。ただし、すべての制約／目的関数をＩＬＰで効果的に表すことができるわけではなく、解は経時的に指数関数的になる可能性がある。ここで説明するヒューリスティックベースのパーティショナは貪欲であり、したがって、経時的に線形である。パーティショニングは、マッピングと並行して行われる。代替的に、パーティショニングとマッピングとを同時に実行することもできる。
【００７０】
図１３は、一例による、カーネルをパーティションに割り当てる方法１３００を示す流れ図である。方法１３００は、上記の方法１２００のステップ１２０８の一部として実行することができる。方法１３００は、ステップ１３０２において開始し、ここで、ＤＰＥアレイコンパイラ６０４は、（例えば、ステップ１２０６から）深さ優先の逆後行順でソートされたノードのセットＰを取得する。ステップ１３０４において、ＤＰＥアレイコンパイラ６０４は、セットＰが空であるか否か（例えば、すべてのカーネルがパーティションに割り当てられているか否か）を決定する。そうである場合、方法１３００は、ステップ１３０６に進み、通信最小化のためにパーティショニング後の最適化を実行する。そうでなければ、方法１３００はステップ１３０８に進む。ステップ１３０８において、ＤＰＥアレイコンパイラ６０４は、ノードｎをセットＰから削除する。ステップ１３１０において、ＤＰＥアレイコンパイラ６０４は、ノードｎを既存のまたは新しいパーティションに割り当てる。
【００７１】
特に、ＤＰＥアレイコンパイラ６０４は、ｎを追加することができる既存のパーティションのセットＲを決定する。ステップ１３１２において、ＤＰＥアレイコンパイラ６０４は、ｎと共有されるバッファの降順でパーティションＲをソートする。上記のように、一部のカーネルは他のカーネルとメモリバッファを共有することができる。ステップ１３１４において、ＤＰＥアレイコンパイラ６０４は、ｎとｒのマージが（１）クリティカルパスの悪化、または（２）実行不可能なトポロジ配置をもたらさないように、ソートされたＲの最初のパーティションｒを選択する。
【００７２】
ステップ１３１６において、ＤＰＥアレイコンパイラ６０４は、ｒが空であるかどうか（すなわち、ｎのための既存のパーティションがないか）を決定する。そうでない場合、方法１３００は、ステップ１３１８に進み、ここで、ＤＰＥアレイコンパイラ６０４は、ノードｎをパーティションｒとマージし、ノードｎに基づいてパーティションｒのロケーション制約（例えば、ｎに対して定義された任意の絶対制約および／または相対制約）を更新する。ｒが空の場合、方法１３００は、代わりにステップ１３２０に進み、そこで、ＤＰＥアレイコンパイラ６０４は、新しいパーティションを作成し、ｎを新しいパーティションに追加し、ｎに基づいて新しいパーティションのロケーション制約を更新する。方法１３００は、ステップ１３１８またはステップ１３２０のいずれかからステップ１３０４に進み、すべてのノードが処理されるまで繰り返される。
【００７３】
<u>マッピング</u>
上記のステップ１１０８で説明したように、ＤＰＥアレイコンパイラ６０４は、カーネルおよびカーネルクラスタをＤＰＥ１１０にマッピングする。マッピングアルゴリズムへの入力は、静的有向グラフ（図９Ｂなど）およびデバイス記述である。マッピングアルゴリズムは、出力として、各カーネルが配置され、カーネル間のバッファが配置され、ＩＯノードがマッピングされるＤＰＥアレイ１０５内の物理的位置を提供する。
【００７４】
図１４は、一例による、カーネルおよびカーネルクラスタをＤＰＥ１１０にマッピングする方法１４００を示す流れ図である。方法１４００は、上記の方法１１００のステップ１

１０８の一部として実行することができる。方法１４００は、ステップ１４０２において開始し、ここで、ＤＰＥアレイコンパイラ６０４は、カーネル間のブロックベースのデータ移動（例えば、ＤＰＥアレイ１０５内のデータ移動）のためのＤＭＡ通信を導入することなく、カーネル、バッファ、およびＩＯノードをマッピングする第１のパスを実行する。ステップ１４０４において、ＤＰＥアレイコンパイラ６０４は、任意のＤＭＡが必要か否かを決定する。必要でない場合、方法１４００は、ステップ１４１０に進み、終了する。必要である場合、方法１４００はステップ１４０６に進む。

【００７５】

ステップ１４０６において、ＤＰＥアレイコンパイラ６０４は、ＤＰＥアレイ１０５内のカーネル間のＤＭＡ通信を提供するために、有向グラフに追加のバッファを挿入する。ステップ１４０８において、ＤＰＥアレイコンパイラ６０４は、ＤＭＡ通信リンクを有するマッピングの第２のパスを実行する。マッピングの第２のパスは、第１のマッピングパスからの解を開始点として使用することができるため、第１のパスよりも高速に実行することができる。したがって、ＤＰＥアレイコンパイラ６０４は、ＤＭＡ通信のために新しく挿入されたバッファを配置するだけでよい。次いで、方法１４００は、ステップ１４１０において終了する。

【００７６】

両方のマッピングパス（１４０２および１４０８）は、ヒューリスティックな費用目標を使用してＩＬＰベースの最適化問題を解決する。最適化問題の目標は、（１）データ移動の数を最小限に抑えること、２）メモリの競合を最小限に抑えること、および、３）待ち時間を最小限に抑えることである。

【００７７】

データ移動の最適化：ＤＰＥ　１１０の各コア２０２は、図３に示すように、すべての基本方位側（北、南、東、西）のメモリモジュール３５１にアクセスすることができる。カーネルをコア２０２にマッピングするとき、ＤＰＥアレイコンパイラ６０４は、カーネルによってアクセスされるすべてのバッファが、特定のコア２０２から直接アクセスされ得るメモリモジュール３５１の１つに配置されることを保証する。異なるコア２０２にマッピングされた２つのカーネルは、２つの異なる方法、すなわち、非ＤＭＡおよびＤＭＡでメモリを介して通信することができる。非ＤＭＡの場合、カーネルはデータをメモリに書き込んでおり、これは別のカーネルによって読み出される。２つのカーネルが、同じメモリモジュール３５１にアクセスすることができるコア２０２にマッピングされる場合、コア２０２間のＤＭＡ通信は必要とされない。ＤＭＡの場合、２つのカーネルは、同じメモリモジュール３５１にアクセスすることができないコア２０２にマッピングされる。そのような場合、カーネル間のバッファは、２つの異なるメモリモジュール３５１に複製され、そのうちの１つは、第１のコア２０２によってアクセスされ得、他方は、第２のコア２０２によってアクセスされ得る。第１のメモリモジュール３５１からのデータは、ＤＭＡを使用してＤＰＥ相互接続２０９を介して第２のメモリモジュール３５１に転送される。非ＤＭＡと比較して、ＤＭＡは、ＤＰＥ相互接続２０９内のルーティングリソースに加えて、２倍のメモリフットプリント、２倍の数のロック、および２つのＤＭＡチャネルを必要とする。

【００７８】

ＤＰＥアレイコンパイラ６０４は、各コアの隣接するメモリモジュールのアクセスコストがゼロであり、残りのメモリモジュールのコストがより高いコストモデルを使用して、ＤＰＥアレイ１０５を格子縞アーキテクチャとしてモデル化する。このとき、最適化問題は、最小限のコストでカーネルをコアに、バッファをメモリモジュールにマッピングすることである。この問題は、二次最適化問題として自然に表現可能である。ＤＰＥアレイコンパイラ６０４は、二次問題をＩＬＰ問題に縮小するように機能する。

【００７９】

メモリ競合最適化：各メモリモジュール３５１は、ＲＡＭバンク３１８（例えば、ＲＡＭの８つのバンク）を含む。同じサイクルで同じＲＡＭバンクに複数のアクセスがある場合

、メモリアクセス競合が発生する。メモリアクセス競合は、以下の種々のタイプ、すなわち、（１）コア内メモリアクセス競合、（２）コア間メモリアクセス競合、（３）コア－ＤＭＡメモリアクセス競合、（４）ＤＭＡ－ＤＭＡメモリアクセス競合に分類することができる。コア内メモリアクセス競合の場合、コアは超長命令語（ＶＬＩＷ）命令を実行する。各ＶＬＩＷ命令は、複数のメモリアクセス命令を有することができる（例えば、最大２つのロードおよび１つのストア）。単一の命令における２つ以上のメモリ動作が同じメモリバンクにアクセスすると、メモリストールが発生し、次にコアストールが発生する。同じサイクルで同じメモリバンクにアクセスする２つ以上の異なるコアは、コア間メモリアクセス競合を引き起こす。コアおよびＤＭＡチャネルが同じサイクルで同じメモリバンクにアクセスすると、コア－ＤＭＡメモリアクセス競合が発生する。複数のＤＭＡチャネルが同じサイクルで同じメモリバンクにアクセスすると、ＤＭＡ－ＤＭＡメモリアクセス競合が発生する。

【００８０】

競合を完全に回避することはすべてのアプリケーションで可能であるとは限らないため、ＤＰＥアレイコンパイラ６０４は、ユーザが競合回避および競合最小化設定のセットから選択することを可能にする。ＤＰＥアレイコンパイラ６０４は、ローカルバッファ（例えば、単一のカーネルによってアクセスされるバッファ）と共有バッファ（例えば、複数のカーネルによってアクセスされるバッファ）とを区別し、異なる最適化を実行する。ＤＰＥアレイコンパイラ６０４は、メモリ競合に対処するために二面アプローチ、すなわち、１）競合回避および２）競合最小化をとる。競合回避の場合、データブロックのプロデューサとコンシューマとの間のアクセス競合を回避するために、ＤＰＥアレイコンパイラ６０４は、二重バッファ（例えば、ｐｉｎｇバッファおよびｐｏｎｇバッファ）が異なるＲＡＭバンクにマッピングされることを保証する。同様に、ＤＰＥアレイコンパイラ６０４は、異なるカーネルからのローカルバッファのアクセス間に競合がないことを、それらを異なるバンクに配置することによって保証する。コア内メモリ競合は、単一のカーネルによってアクセスされるすべてのバッファを異なるＲＡＭバンク３１８に配置することによって回避される。

【００８１】

競合最小化の場合、問題は、所与のメモリバンクにアクセスしている独立したアクタ（コア、ＤＭＡチャネル）の数を最小化するという問題に還元される。これをＩＬＰ問題としてモデル化すると、コアおよびＤＭＡチャネルの数はｃ×ｒに比例するため、大規模なデバイスではコストがかかる可能性がある。ここで、ｃはデバイスのＤＰＥアレイ１０５の列の数であり、ｒは行の数である。ＤＰＥアレイコンパイラ６０４は、すべてのＤＭＡチャネルを、ｃ　ｘ　ｒ　ｘ　４の異なるエンティティではなく、２つの個別のアクタ（リーダおよびライタ）としてモデル化することによって、ＩＬＰ変数の数を低減する技法を採用する。

【００８２】

待ち時間最小化：ＦＰＧＡ配置アルゴリズムと同様に、ＤＰＥアレイコンパイラ６０４は、ソースとシンクとの間の距離を最小化することによって、ストリームベースの通信の待ち時間を最小化する。

【００８３】

異種マルチコアアーキテクチャにおけるストリームＦＩＦＯ挿入

デッドロック回避およびパフォーマンスのためのＦＩＦＯの決定および挿入は過去に研究されてきたが、主に計算の理論モデル（同期データフロー、カーンプロセスネットワークなど）および高位合成のコンテキストにおいてであった。この問題はマルチプロセッサシステムでは解決されていない。これは主に、弾性ハンドシェイクストリームを使用して互いに通信するようなシステムがほとんどないためである（例えば、ほとんどのマルチプロセッサシステムは、データ通信に共有メモリを使用するか、またはロックステップにおいて作動するシストリックアレイである）。

【００８４】

図１５は、異種マルチコアアーキテクチャのアプリケーションを実装する際のＦＩＦＯ挿入の方法１５００を示す流れ図である。方法１５００は、上記の方法１１００のステップ１１０８、１１１０、１１１２、１１１４、および１１１６のいずれかの最中に実行することができる。方法１５００は、ステップ１５０２において開始し、ここで、ＤＰＥアレイコンパイラ６０４は、再収束計算および通信パスに沿ったストリームデータ待ち時間の不一致に起因して、アプリケーションがデッドロックであるかまたはパフォーマンス基準を満たさないかを決定する。障害が発生した場合（ステップ１５０４）、方法１５００はステップ１５０６に進む。障害が発生しなかった場合、方法１５００はステップ１５１６において終了する。ステップ１５０６において、ＤＰＥアレイコンパイラ６０４は、計算および通信パスで識別されたデッドロックおよび／またはパフォーマンス障害を回避するための最小ＦＩＦＯサイズを決定する。一例では、１つまたは複数の最小ＦＩＦＯサイズを事前定義する（例えば、ユーザによって指定する）ことができる。

【００８５】

一般的なケースでこれを理論的に分析することは難しく、保守的であり、これによって、ＦＩＦＯが非常に大きくなる可能性がある。したがって、一例では、ＤＰＥアレイコンパイラ６０４は、シミュレーションベースのアプローチを実施する。ＤＰＥアレイコンパイラ６０４は、デッドロック／パフォーマンス障害が回避されるまで、選択されたＦＩＦＯサイズでシステムをシミュレートする（ステップ１５０８）。シミュレーションは、様々な抽象化レベルで行うことができる、すなわち、計算カーネルコードは非時限であり得るが、カーネルは同時に実行され（「カーネルの非時限および同時実行」）、または、カーネルはサイクルが正確にモデル化され得る（「カーネルのサイクルが正確な同時実行」）。

【００８６】

ステップ１５０６においてＦＩＦＯサイズが決定されると、ＦＩＦＯは、プロデューサ計算カーネルとコンシューマ計算カーネルとの間のストリームルートに沿って挿入される必要がある（ステップ１５１０）。ＤＰＥアレイ１０５には、２つのオプションがある、すなわち、各ストリームスイッチが、制限されたサイズ（例えば、各々１６語）の２つのＦＩＦＯを有するか、または、ローカルデータメモリがタイルＤＭＡエンジンを介してＦＩＦＯとして使用され得る。ＦＩＦＯサイズが非常に大きい場合は、後者のオプションが必要である。前者のオプションには、プロデューサからコンシューマへのルートに沿ったストリームスイッチの数によって、使用することができるサイズ制限付きのＦＩＦＯの総数が制限されるという１つの問題がある。したがって、指定または決定されたＦＩＦＯサイズの合計を満たすために、ルート自体を人為的に長くする必要がある場合がある。もう１つの問題は、複数のルートがストリームスイッチを共有する可能性があることである。したがって、ルートのストリームスイッチに沿って指定または決定されたＦＩＦＯの長さを分散するヒューリスティックは、そのような共有を考慮する。したがって、ステップ１５１２において、ＤＰＥアレイコンパイラ６０４は、ＤＰＥ相互接続内のＦＩＦＯを選択することができる。加えて、または代替的に、ステップ１５１４において、ＤＰＥアレイコンパイラ６０４は、ローカルメモリにおいてＦＩＦＯをｊ実装することができる。次いで、方法１５００は、ステップ１５１６において終了する。

【００８７】

図１６は、一例による処理システム１６００を示すブロック図である。処理システム１６００は、上述のステップ１５０６の結果とすることができる。この例では、処理システム１６００は、データソース１６０２と、複数のＤＰＥ　１６０４Ａ～１６０４Ｄ）とを含む。データソース１６０２は、各ＤＰＥ　１６０４によって処理されるデータを提供する。ＤＰＥ　１６０４は直列に結合される（例えば、ＤＰＥ　１６０４Ａ、ＤＰＥ　１６０４Ｂ、ＤＰＥ　１６０４Ｃ、ＤＰＥ　１６０４Ｄによってこの順で形成されたパイプライン）。ステップ１５０６において、ＤＰＥアレイコンパイラ６０４は、データソース１６０２とＤＰＥ　１６０４Ａとの間のパスがＦＩＦＯを必要としないこと、データソース１６０２とＤＰＥ　１６０４Ｂとの間のパスが深さ１０のＦＩＦＯを必要とすること、デー

タソース１６０２とＤＰＥ　１６０４Ｃとの間のパスが深さ２０のＦＩＦＯを必要とすること、および、データソース１６０２とＤＰＥ　１６０４Ｄとの間のパスが深さ３０のＦＩＦＯを必要とすることを決定することができる。

【００８８】

図１７Ａは、一例による処理システム１６００の実施態様１７００を示すブロック図である。この例では、実施態様１７００は、ノード１７０２、１７０４、および１７０６を含み、それらの各々がＤＰＥ相互接続２０９内のスイッチを表す。実施態様１７００は、最も多いＦＩＦＯリソース（例えば、深さ１０、２０、および３０、合計６０の深さのＦＩＦＯ）を必要とするため、処理システム１６００の最悪の場合の実施態様である。したがって、ステップ１５１０において、ＤＰＥアレイコンパイラ６０４は、下記のアルゴリズムにおいて説明するように、ＦＩＦＯのより効率的な配置を実行することができる。

【００８９】

図１７Ｂは、別の例による処理システム１６００の実施態様１７０１を示すブロック図である。図１７Ｂの例では、実施態様１７０１は、深さ１０、１０、および２０、合計深さ４０のＦＩＦＯを含む。実施態様１７０１は、実施態様１７００よりも少ないリソースを使用して処理システム１６００の必要なＦＩＦＯを達成する。

【００９０】

図１８は、一例による、ＦＩＦＯを配置する方法１８００を示す流れ図である。方法１８００は、方法１５００のステップ１５１０においてＤＰＥアレイコンパイラ６０４によって実行することができる。方法１８００は、ステップ１８０２において開始し、ＤＰＥアレイコンパイラ６０４がパス順序を決定する。パスは、データソース１６０２とＤＰＥ　１６０４との間の１つまたは複数のノードの集合である。一例では、ＤＰＥアレイコンパイラ６０４は、任意の順序において（例えば、左から右へ）パスを処理する。別の例を下記にさらに説明する。ステップ１８０４において、ＤＰＥアレイコンパイラ６０４は、各パスに沿ったノード順序を決定する。一例では、ＤＰＥアレイコンパイラ６０４は、ノード順序をＤＰＥ　１６０４からデータソース１６０２に向かうように決定する。別の例を下記にさらに説明する。

【００９１】

ステップ１８１０において、ＤＰＥアレイコンパイラ６０４は、処理すべきパスを選択する。ステップ１８１２において、ＤＰＥアレイコンパイラ６０４は、実行可能なＦＩＦＯ配置に達するまで、ノード順序に沿ってＦＩＦＯ要件でエッジに注釈を付ける。場合によっては、ＤＰＥアレイコンパイラ６０４は、ＦＩＦＯ要件を依然として満たしながら実現可能性に達するために可能な限り多くの共通ＦＩＦＯを「リタイム」する（ステップ１８１４）。例えば、データソース１６０２とＤＰＥ　１６０４Ｃとの間のパスを考える。このパスを処理するとき、ＤＰＥアレイコンパイラ６０４は、ノード１７０４と１７０６との間に深さ１０のＦＩＦＯを割り当て、ノード１７０６とＤＰＥ　１６０４Ｃとの間に深さ１０のＦＩＦＯを割り当てることができる。これは、ＤＰＥ　１６０４Ｂおよび１６０４Ｃの両方のＦＩＦＯ要件を満たす。しかしながら、データソース１６０２とＤＰＥ　１６０４Ｄとの間のパスを処理するとき、ＤＰＥアレイコンパイラ６０４は、データソース１６０２とノード１７０４との間のＦＩＦＯを０の深さから１０の深さまでリタイムし、ノード１７０４と１７０６との間のＦＩＦＯを１０の深さから０の深さまでリタイムすることができる。結果を図１７Ｂに示す。ＤＰＥアレイコンパイラ６０４は、ＦＩＦＯ要件を満たすようにノード１７０４とＤＰＥ　１７０４Ｄとの間の深さ２０のＦＩＦＯを決定する。

【００９２】

一例では、ステップ１８０２において、ＤＰＥアレイコンパイラ６０４は、処理されるパスの順序を決定するために全グラフ解析を実行する。ＦＩＦＯを割り当てるために一度に１つのパスを調べる代わりに、ＤＰＥアレイコンパイラ６０４は、ＦＩＦＯ挿入を必要とするすべてのパスを調べることができる。次に、ＤＰＥアレイコンパイラ６０４は、パスに沿ったノードの数に関して、パスのサイズの昇順でパスをソートすることができる（ス

テップ１８０４）。パスが等しい数のノードを有する場合、ＤＰＥアレイコンパイラ６０４は、パスのＦＩＦＯ深度に基づいて、最小深度から最大深度の順にソートすることができる。

【００９３】

一例では、ステップ１８０６において、ＤＰＥアレイコンパイラ６０４は、ステップ１８０４において識別された各パスに沿ってノードを並べ替える。ノードの次数は、ノードがすべてのパスにわたって使用された合計回数として定義される。ＤＰＥアレイコンパイラ６０４は、ノードを次数の昇順にソートすることができる。ＦＩＦＯ判定の前に全グラフ解析およびノード順序付けを実行することによって、方法１８００は、ＤＰＥの近くの深さを更新することが可能であると同時に、共通ＦＩＦＯ深さをデータソースに向かって移動させることができる。さらに、方法１８００は、ＦＩＦＯポートからのブロードキャストを処理することができる。

【００９４】

図１７Ａの例では、ＦＩＦＯを必要とする最短パスは、データソース１６０２とＤＰＥ　１６０４Ｄとの間である。データソース１６０２とＤＰＥ　１６０４Ｂとの間のパスおよびデータソース１６０２と１６０４Ｃとの間のパスは、同じ長さ（例えば、３つのノード）である。しかしながら、データソース１６０２とＤＰＥ　１６０４Ｂとの間のパスは、データソース１６０２とＤＰＥ　１６０４Ｃとの間のパス（例えば、２０）に対してより浅いＦＩＦＯ深さ（例えば、１０）を有する。したがって、ＤＰＥアレイコンパイラ６０４は、データソース１６０２とＤＰＥ　１６０４Ｄとの間のパスを処理し、続いてまずデータソース１６０２とＤＰＥ　１６０４Ｂとの間のパスを処理し、最後にデータソース１６０２とＤＰＥ　１６０４Ｃとの間のパスを処理することができる。

【００９５】

図１７Ａの例では、ノード１７０２は３つのパスの一部であり、ノード１７０４は３つのパスの一部であり、ノード１７０６は２つのパスの一部である。したがって、データソースとＤＰＥ１６０４Ｂおよび１６０４Ｃとの間のパスについて、ノード順序は１７０６、１７０４、および１７０２である。データソース１６０２とＤＰＥ　１６０４Ｄとの間のパスについて、ノード順序は１７０４および１７０２である。

【００９６】

次に、ＤＰＥアレイコンパイラ６０４は、決定された順序で、各パスについて決定されたノード順序でパスを処理することによってステップ１８１０～１８１６を実行する。結果は図１７Ｂに示されており、これはこの例における前の例（パスソートおよびノード並べ替えなし）からの結果と同じである。しかしながら、実際の例では、得られる解は異なり得る。

【００９７】

ＤＰＥアレイ内のコア間の接続のルーティング

ＤＰＥアレイ内のコア間のルーティングは、ＰＬ　１２２との通信を必要とするルートにチャネルを貪欲に配分することによって達成することができる。貪欲型ヒューリスティックであるため、この手法は、より大きなグラフをルーティングするとき、または特別な制約を処理する必要があるときに制限を明らかにする。従来の手法は、プログラミングモデルにおける明示的なパケット交換のためのアップサイズ／ダウンサイズ変換およびルーティングを必要とするアーキテクチャ上の制約、パケット交換、およびチャネルの処理をサポートしていない。本明細書では、これらの要件を処理するルーティングのための技法について説明する。

【００９８】

以下の用語は、ＤＰＥアレイ１０５にマッピングされたアプリケーションをルーティングするための本明細書に記載のルーティング技法を説明する際に使用するために導入される。ルーティングノード：データのソースもしくは宛先または中間スイッチを表すルーティンググラフ内のノード。ノード容量：ノードを通る最大許容データフローを表す整数。ルーティングエッジ：ルーティングエッジは、ソースから宛先への可能なデータの流れを表

す。ルーティンググラフ：ルーティンググラフは、すべての可能なルーティング選択肢を表す。これらの選択肢は、アーキテクチャ切り替え制約、ユーザ定義のシム制約によって課されるルーティング制約、チャネルのアップサイジング／ダウンサイジングの制約、ならびに、明示的なパケット分割およびマージ動作によるプログラマ定義の制約をモデル化する。ネット：ネットは、ルーティンググラフ内のソースノードおよびルーティンググラフ内の複数の宛先を用いた所望のルーティングを表す。ネット利用率：ネットによって必要とされる帯域幅をモデル化する整数。低帯域幅のネットは、スイッチングリソースを共有することによってともにルーティングされ得る。

【００９９】

図１９は、一例による、ＤＰＥアレイ１０５にマッピングされたアプリケーションをルーティングする方法１９００を示す流れ図である。方法１９００は、上述の方法１１００のステップ１１１６中に実行することができる。方法１９００は、ステップ１９０２において開始し、ＤＰＥアレイコンパイラ６０４がルーティンググラフを構築する。ＤＰＥアレイコンパイラ６０４は、ＤＰＥアレイ１０５アーキテクチャの記述に基づいてルーティンググラフを構築する。各ストリームスイッチポートは、単一のルーティングノードとしてモデル化される。各ＤＭＡ、制御ポート、およびコアストリームポートは、ルーティングノードとしてモデル化される。スイッチングのための可能な選択は、ノード間のエッジとして表される。アーキテクチャにおいて許容されるすべての可能なスイッチングオプションが、ルーティンググラフにおいてモデル化される。ルーティンググラフを構築するには、ＤＰＥアレイ１０５のアーキテクチャの記述のみが必要である。

【０１００】

ステップ１９０４において、ＤＰＥアレイコンパイラ６０４は、ＰＬ接続をモデル化する。ＰＬノードの出力はいずれのシムポートにも接続することができるため、ＤＰＥアレイコンパイラ６０４は、各ＰＬ接続をＰＬソースからシム内のすべてのチャネルへのクロスバー接続としてモデル化する。ユーザがシムチャネルに対して特定の制約を指定する場合、クロスバー接続は、所与のシム制約のセットに特化することができる。

【０１０１】

ステップ１９０６において、ＤＰＥアレイコンパイラ６０４は、アップサイザ／ダウンサイザ接続をモデル化する。シムアーキテクチャは、より低い周波数で実行されるより高いビット幅のチャネルが、より高い周波数で実行されるより低いビット幅のチャネルに接続されることを可能にする。シムチャネルは固定ビット幅を有し、そのため、いくつかのより高いビット幅のチャネルを実装するには、複数の隣接するシムチャネルを使用する必要がある。このアーキテクチャは、シムチャネルのグループを偶数境界に配分しなければならないという制限をさらに課す。これらの制約は、新しいノードおよびエッジでルーティンググラフを修正することによって組み込まれる。制約は、クロスバー接続を、接続性が制限されたすべてのシムチャネルに置き換えることによって表される。

【０１０２】

ステップ１９０８において、ＤＰＥアレイコンパイラ６０４は、他の制約をモデル化する。いくつかの制約は、ルーティンググラフにおける接続として容易に表現可能でない。これらは、ネットおよびリソースルートに対する追加のグローバル制約として表される。例えば、アーキテクチャ制約は、４つのパケットスイッチネットがすべてのスイッチポートを通過することを可能にすることであり得る。別の例は、帯域幅利用率が低い場合でも、１つのネットのみがシムチャネルを通過できるようにすることである。早期または後期の明示的なパケット交換ノードのマージは、制約メカニズムを使用して処理される。

【０１０３】

ステップ１９１０において、ＤＰＥアレイコンパイラ６０４は、充足可能性ソルバ（ＳＡＴソルバ）を呼び出して、アプリケーションをＤＰＥアレイ１０５内でルーティングする。入力仕様グラフが、ルーティングを必要とするネットについて検査される。ネットのソースおよび宛先が識別される。ソースまたは宛先ネットは、ルーティンググラフ内のノードでなければならない。パケット交換を可能にするために、ネット利用率がユーザによっ

て提供される。入力仕様内のすべてのネットが、制約に沿ってＳＡＴソルバに共に渡される。ＳＡＴソルバによって提供される解は、ＤＰＥアレイ１０５のストリームスイッチ（例えば、ストリームスイッチ構成コード６１６）をプログラムするために使用される。

【０１０４】

図２０は、図１に示されるデバイス１００の一実施態様として使用することができる、一例によるプログラム可能なＩＣ１を示すブロック図である。プログラマブルＩＣ　１は、プログラマブルロジック３と、構成ロジック２５と、構成メモリ２６とを含む。プログラマブルＩＣ　１は、不揮発性メモリ２７、ＤＲＡＭ　２８、および他の回路２９などの外部回路に結合され得る。プログラマブルロジック３は、論理セル３０と、サポート回路３１と、プログラマブル相互接続３２とを含む。論理セル３０は、複数の入力の一般的な論理機能を実装するように構成することができる回路を含む。サポート回路３１は、トランシーバ、入出力ブロック、デジタル信号プロセッサ、メモリなどの専用回路を含む。論理セルおよびサポート回路３１は、プログラマブル相互接続３２を使用して相互接続することができる。論理セル３０をプログラムし、サポート回路３１のパラメータを設定し、プログラマブル相互接続３２をプログラムするための情報は、構成ロジック２５によって構成メモリ２６に記憶される。構成ロジック２５は、不揮発性メモリ２７または任意の他のソース（例えば、ＤＲＡＭ　２８、または他の回路２９から）から構成データを取得することができる。いくつかの例では、プログラマブルＩＣ　１は処理システム２を含む。処理システム２は、マイクロプロセッサ（複数可）、メモリ、サポート回路、ＩＯ回路などを含むことができる。

【０１０５】

図２１は、トランシーバ３７、構成可能論理ブロック（「ＣＬＢ」）３３、ランダムアクセスメモリブロック（「ＢＲＡＭ」）３４、入出力ブロック（「ＩＯＢ」）３６、構成・クロッキングロジック（「ＣＯＮＦＩＧ／ＣＬＯＣＫＳ」）４２、デジタル信号処理ブロック（「ＤＳＰ」）３５、特殊入出力ブロック（「Ｉ／Ｏ」）４１（例えば、構成ポートおよびクロックポート）、および、デジタルクロックマネージャ、アナログ／デジタル変換器、システム監視ロジックなどの他のプログラマブルロジック３９を含む、多数の異なるプログラマブルタイルを含むプログラマブルＩＣ１のフィールドプログラマブルゲートアレイ（ＦＰＧＡ）実施態様を示す。ＦＰＧＡはまた、ＰＣＩｅインターフェース４０、アナログ－デジタル変換器（ＡＤＣ）３８などを含むことができる。

【０１０６】

いくつかのＦＰＧＡでは、各プログラマブルタイルは、図２１の上部に含まれる例によって示されるように、同じタイル内のプログラマブル論理要素の入力および出力端子４８への接続を有する少なくとも１つのプログラマブル相互接続要素（「ＩＮＴ」）４３を含むことができる。各プログラマブル相互接続要素４３はまた、同じタイルまたは他のタイル（複数可）内の隣接するプログラマブル相互接続要素（複数可）の相互接続セグメント４９への接続を含むことができる。各プログラマブル相互接続要素４３はまた、論理ブロック（図示せず）間の一般的なルーティングリソースの相互接続セグメント５０への接続を含むことができる。一般的なルーティングリソースは、相互接続セグメント（例えば、相互接続セグメント５０）のトラックを備える論理ブロック（図示せず）と、相互接続セグメントを接続するためのスイッチブロック（図示せず）との間のルーティングチャネルを含むことができる。一般的なルーティングリソースの相互接続セグメント（例えば、相互接続セグメント５０）は、１つまたは複数の論理ブロックにまたがることができる。プログラマブル相互接続要素４３は、一般的なルーティングリソースと共に、図示のＦＰＧＡのプログラマブル相互接続構造（「プログラマブル相互接続」）を実装する。

【０１０７】

例示的な実施態様において、ＣＬＢ３３は、ユーザ論理と単一のプログラマブル相互接続要素（「ＩＮＴ」）４３とを実装するようにプログラムすることができる構成可能論理素子（「ＣＬＥ」）４４を含むことができる。ＢＲＡＭ３４は、１つまたは複数のプログラム可能な相互接続要素に加えて、ＢＲＡＭ論理要素（「ＢＲＬ」）４５を含むことができ

る。典型的には、タイルに含まれる相互接続要素の数は、タイルの高さに依存する。図示の例は、ＢＲＡＭタイルは５つのＣＬＢと同じ高さを有するが、他の数（例えば４つ）も使用することができる。ＤＳＰタイル３５は、適切な数のプログラマブル相互接続要素に加えて、ＤＳＰ論理要素（「ＤＳＰＬ」）４６を含むことができる。ＩＯＢ３６は、例えば、プログラマブル相互接続要素４３の１つのインスタンスに加えて、入出力論理素子（「ＩＯＬ」）４７の２つのインスタンスを含むことができる。当業者には明らかなように、例えばＩ／Ｏ論理素子４７に接続される実際のＩ／Ｏパッドは、典型的には、入出力論理素子４７の領域に限定されない。

【０１０８】

図示された例では、ダイの中心付近の水平領域（図１２に示す）が、構成、クロック、および他の制御ロジックに使用される。この水平領域または列から延伸する垂直列５１は、ＦＰＧＡの幅全体にわたってクロックおよび構成信号を分配するために使用される。

【０１０９】

図２１に示されるアーキテクチャを利用するいくつかのＦＰＧＡは、ＦＰＧＡの大部分を構成する規則的な柱状構造を乱す付加的な論理ブロックを含む。追加の論理ブロックは、プログラマブルブロックおよび／または専用ロジックとすることができる。

【０１１０】

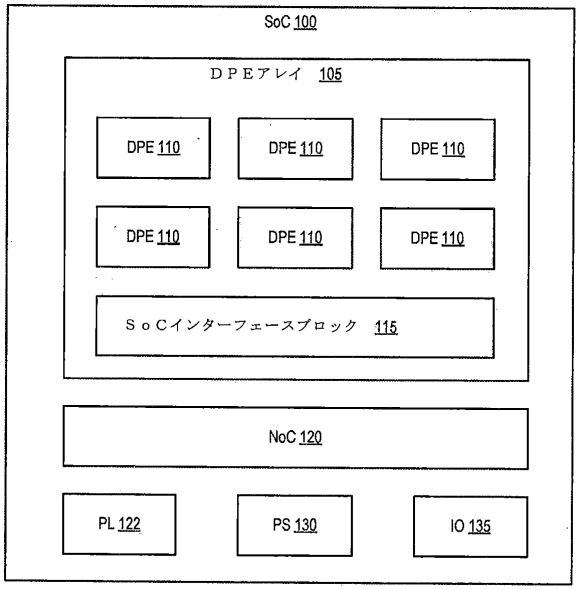
図２１は、例示的なＦＰＧＡアーキテクチャのみを示すことを意図していることに留意されたい。例えば、図２１の上部に含まれる行内の論理ブロックの数、行の相対幅、行の数および順序、行に含まれる論理ブロックのタイプ、論理ブロックの相対サイズ、ならびに、相互接続／論理実施態様は純粋に例示的なものである。例えば、実際のＦＰＧＡでは、ユーザ論理の効率的な実装を容易にするために、ＣＬＢが現れる場所にはどこでも、ＣＬＢの隣接する２つ以上の行が一般に含まれているが、隣接するＣＬＢ行の数はＦＰＧＡの全体サイズによって異なる。

【０１１１】

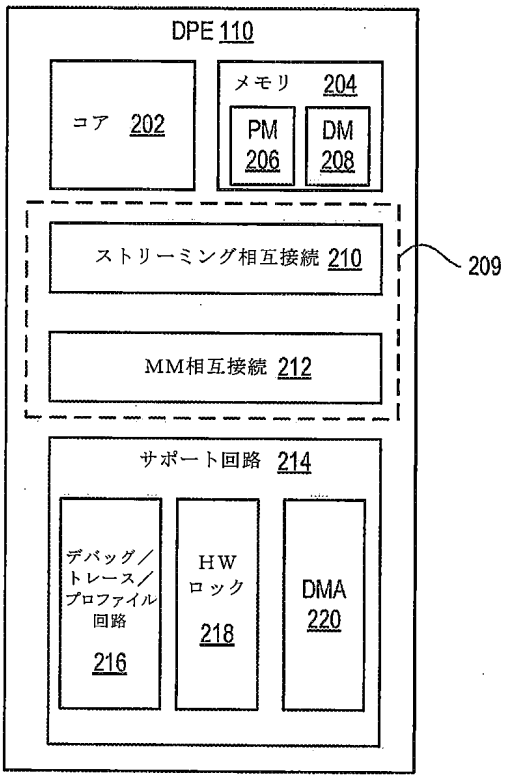
上記は特定の例を対象としているが、その基本的な範囲から逸脱することなく他のおよびさらなる例を考案することができ、その範囲は添付の特許請求の範囲によって決定される。

【図面】

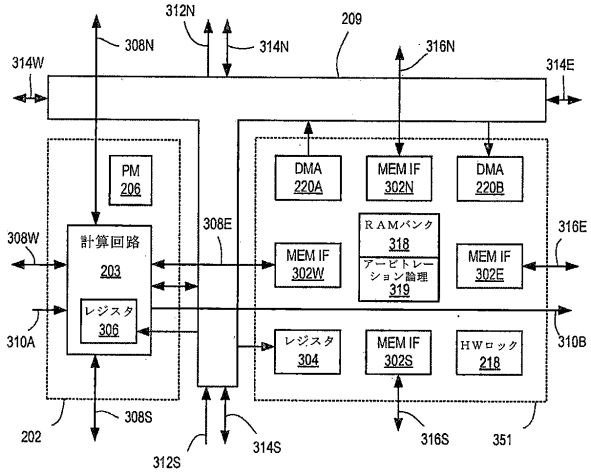
【図１】


SoC 100
ＤＰＥアレイ 105
DPE 110
DPE 110
DPE 110
DPE 110
DPE 110
DPE 110
ＳｏＣインターフェースブロック 115
NoC 120
PL 122
PS 130
IO 135

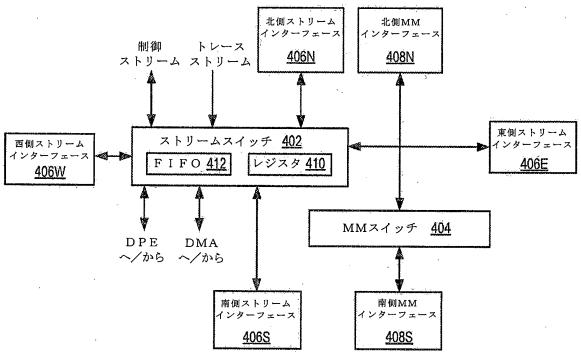
【図２】


DPE 110
コア 202
メモリ 204
PM 206
DM 208
ストリーミング相互接続 210
209
MM相互接続 212
サポート回路 214
デバッグ／トレース／プロファイル回路 216
HW ロック 218
DMA 220

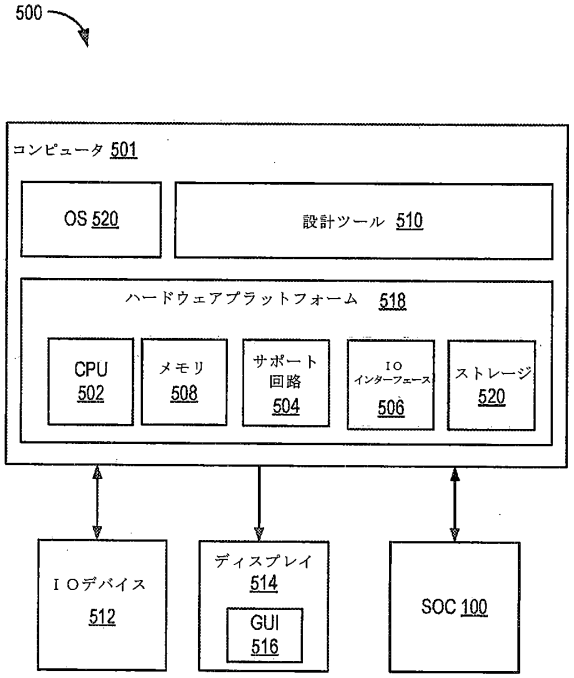
【図３】


312N
308N
314N
209
316N
314W
314E
PM 206
DMA 220A
MEM IF 302N
DMA 220B
RAMバンク 318
308E
316E
308W
計算回路 203
MEM IF 302W
アービトレーション論理 319
MEM IF 302E
レジスタ 306
310A
レジスタ 304
MEM IF 302S
HWロック 218
310B
308S
202
312S
314S
316S
351

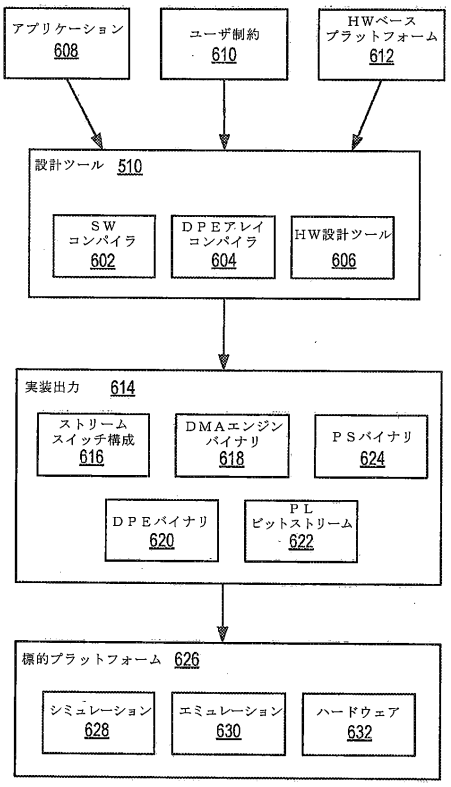
【図４】


制御ストリーム
トレースストリーム
北側ストリームインターフェース 406N
北側MMインターフェース 408N
西側ストリームインターフェース 406W
ストリームスイッチ 402
FIFO 412
レジスタ 410
東側ストリームインターフェース 406E
DPE へ／から
DMA へ／から
MMスイッチ 404
南側ストリームインターフェース 406S
南側MMインターフェース 408S

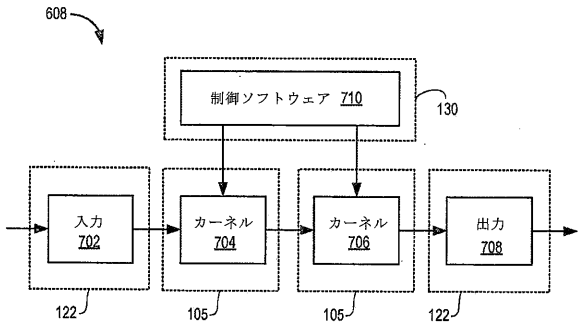
【図５】


500
コンピュータ 501
OS 520
設計ツール 510
ハードウェアプラットフォーム 518
CPU 502
メモリ 508
サポート回路 504
IO インターフェース 506
ストレージ 520
IOデバイス 512
ディスプレイ 514
GUI 516
SOC 100

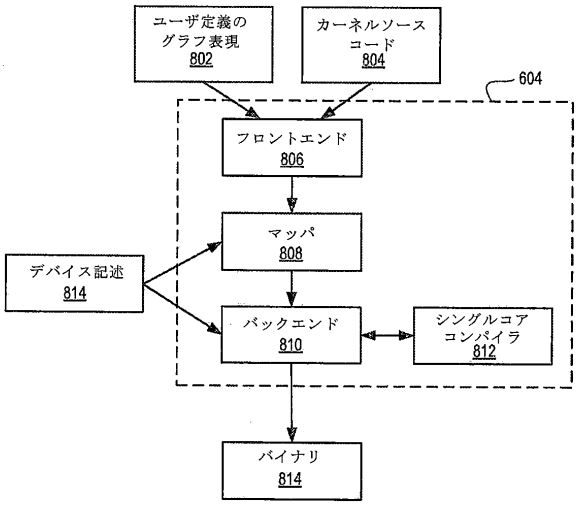
【図６】


アプリケーション 608
ユーザ制約 610
HWベースプラットフォーム 612
設計ツール 510
SW コンパイラ 602
DPEアレイ コンパイラ 604
HW設計ツール 606
実装出力 614
ストリームスイッチ構成 616
DMAエンジンバイナリ 618
PSバイナリ 624
DPEバイナリ 620
PL ビットストリーム 622
標的プラットフォーム 626
シミュレーション 628
エミュレーション 630
ハードウェア 632

【図７】


608
制御ソフトウェア 710
130
入力 702
カーネル 704
カーネル 706
出力 708
122
105
105
122

【図８】


ユーザ定義のグラフ表現 802
カーネルソースコード 804
604
フロントエンド 806
マッパ 808
デバイス記述 814
バックエンド 810
シングルコアコンパイラ 812
バイナリ 814

【図９Ａ】

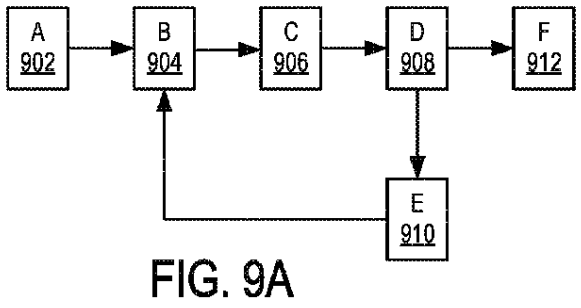


FIG. 9A

【図９Ｂ】

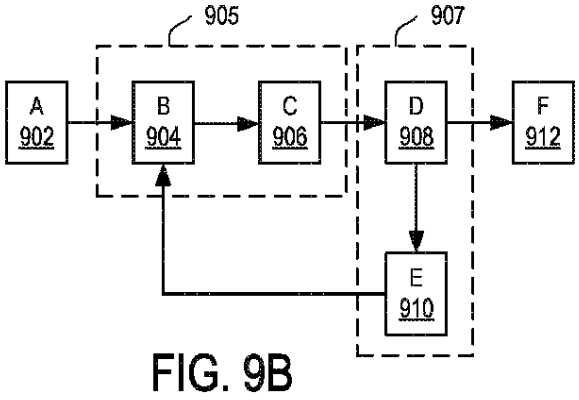


FIG. 9B

【図９Ｃ】

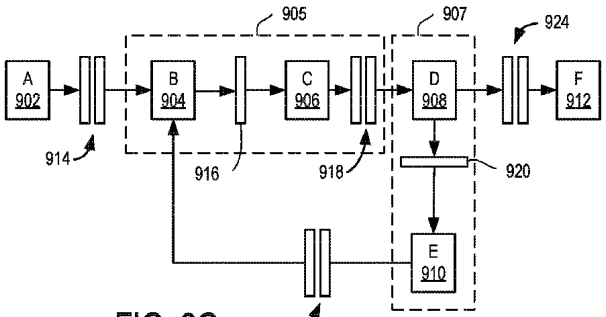


FIG. 9C

【図９Ｄ】

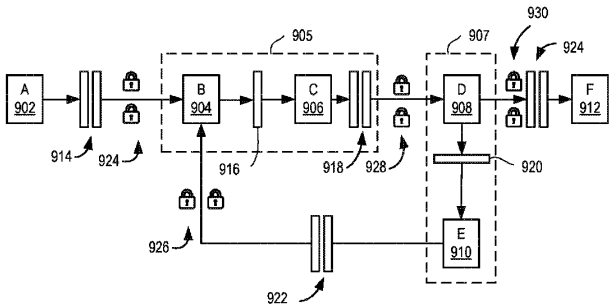
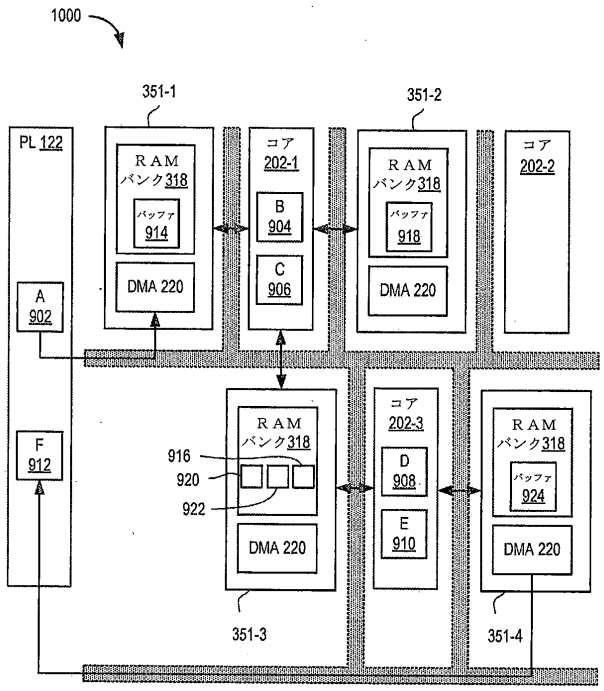
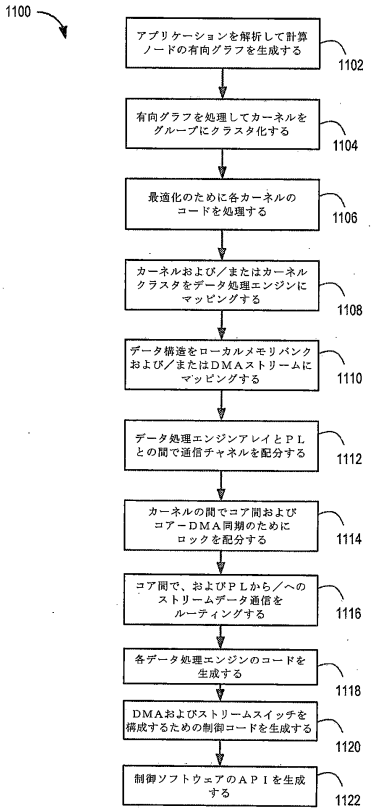


FIG. 9D

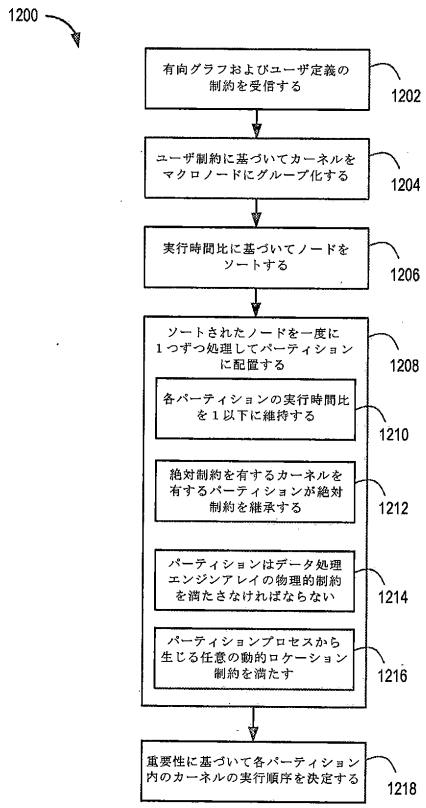
【図１０】



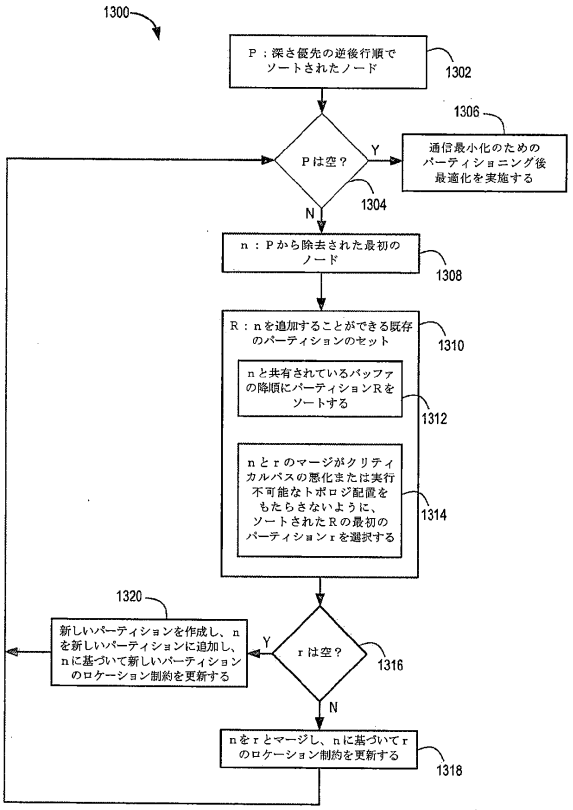
【図１１】



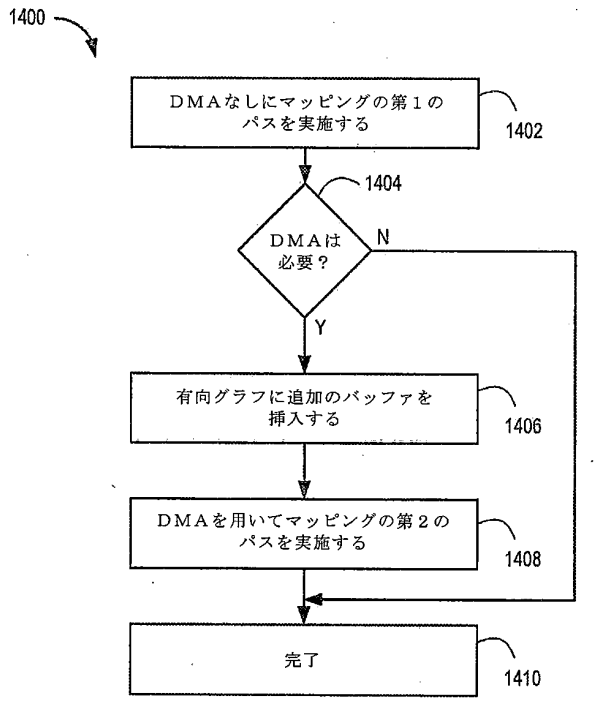
【図１２】



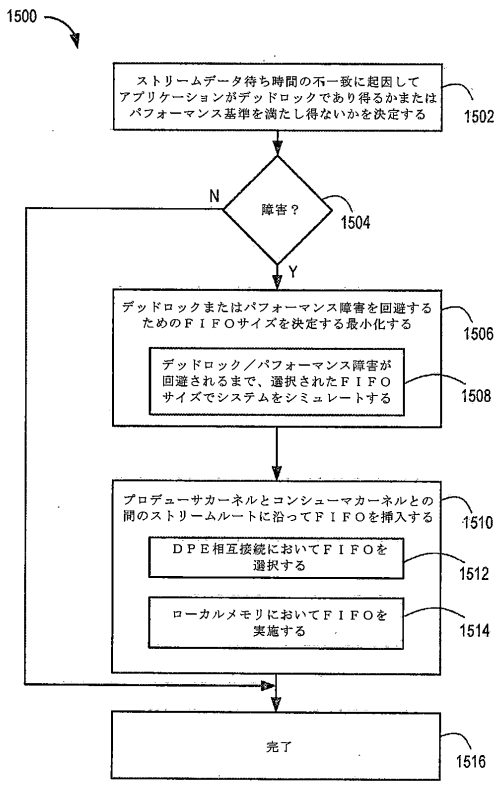
【図１３】



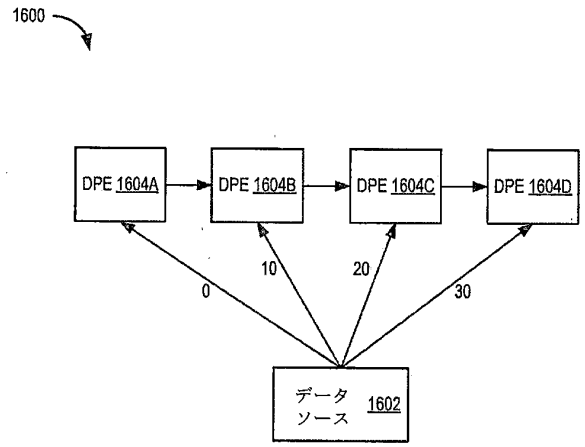
【図１４】



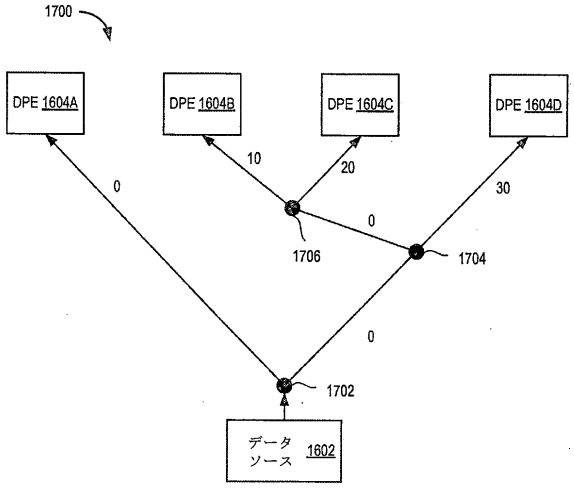
【図１５】



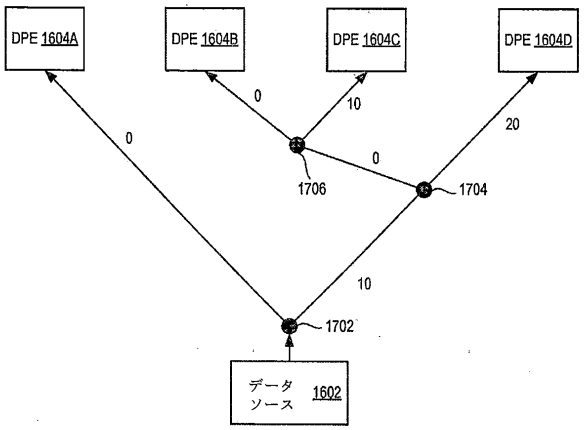
【図１６】



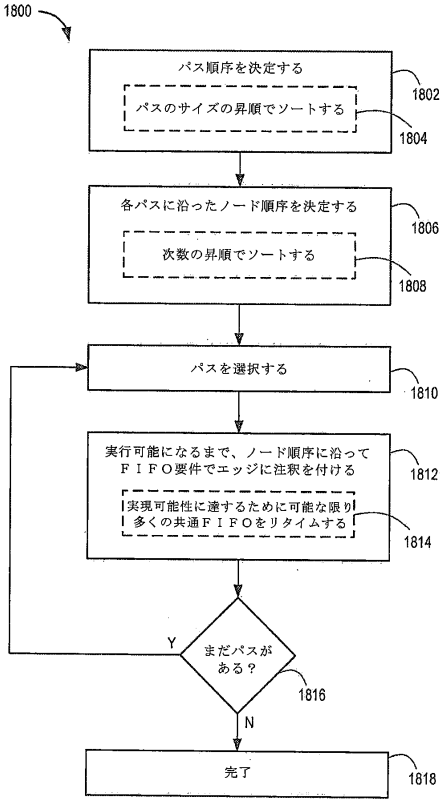
【図１７Ａ】



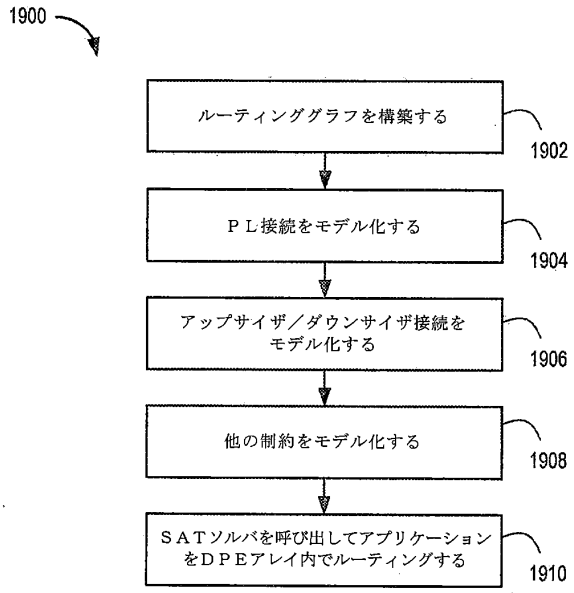
【図１７Ｂ】



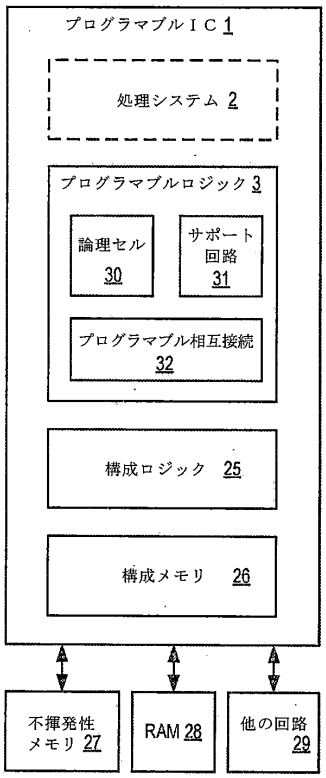
【図１８】



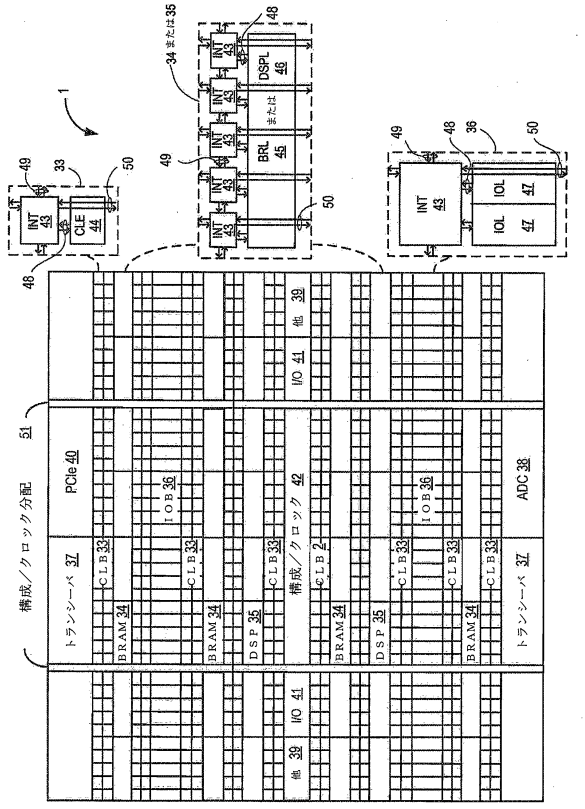
【図１９】



【図２０】



【図２１】


1
34 または 35
48
49
33
50
INT 43
CLE 44
DSPL 46
または
BRL 45
IOL 47
構成／クロック分配 51
PCIe 40
ADC 38
トランシーバ 37
CLB 32
IOB 36
BRAM 34
DSP 35
構成／クロック 42
I/O 41
他 39

【国際調査報告】

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2020/031951

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F30/34 G06F30/30
ADD. G06F115/02

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
EPO-Internal, COMPENDEX, INSPEC, IBM-TDB

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	ROMAIN BRILLU ET AL: "FlexTiles", RAPID SIMULATION AND PERFORMANCE EVALUATION, ACM, 2 PENN PLAZA, SUITE 701 NEW YORK NY 10121-0701 USA, 22 January 2014 (2014-01-22), pages 1-8, XP058035363, DOI: 10.1145/2555486.2555489 ISBN: 978-1-4503-2471-7 the whole document page 4, column 1, paragraph 3 page 2, column 2, last paragraph page 3, column 2, paragraphs 3,4; figure 5 page 4, column 1, last paragraph - column 2, paragraph 1 page 4, column 2, paragraphs 2, 3 page 2, column 2, paragraph 4 ----- -/--	1-15

[X] Further documents are listed in the continuation of Box C. [ ] See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
20 July 2020	31/07/2020

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Wellisch, J

Form PCT/ISA/210 (second sheet) (April 2005)

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2020/031951

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	JAUME BOSCH ET AL: "Exploiting Parallelism on GPUs and FPGAs with OmpSs", PROCEEDINGS OF THE 1ST WORKSHOP ON AUTOTUNING AND ADAPTIVITY APPROACHES FOR ENERGY EFFICIENT HPC SYSTEMS , ANDARE '17, 1 January 2017 (2017-01-01), pages 1-5, XP055716069, New York, New York, USA DOI: 10.1145/3152821.3152880 ISBN: 978-1-4503-5363-2 the whole document abstract page 1, column 2, paragraph 1 page 2, column 1, paragraph 4 - page 3, column 2, paragraph 3 -----	1-15
X	INGRID VERBAUWHEDE ET AL: "Architectures and Design techniques for energy efficient embedded DSP andmultimedia processing", ACM, 2 PENN PLAZA, SUITE 701 - NEW YORK USA, 1 January 2004 (2004-01-01), pages 1-6, XP040174347, DATE 2004 the whole document page 4, column 2, paragraph 2 - paragraph 4 -----	1-15

Form PCT/ISA/210 (continuation of second sheet) (April 2005)

フロントページの続き

(51)国際特許分類	FI	テーマコード（参考）
G06F 115/02 (2020.01)	G06F 15/173 665D	
	G06F 30/347	
	G06F 115:02	

MK,MT,NL,NO,PL,PT,RO,RS,SE,SI,SK,SM,TR),OA(BF,BJ,CF,CG,CI,CM,GA,GN,GQ,GW,KM,ML,MR,NE,SN,TD,TG),AE,AG,AL,AM,AO,AT,AU,AZ,BA,BB,BG,BH,BN,BR,BW,BY,BZ,CA,CH,CL,CN,CO,CR,CU,CZ,DE,DJ,DK,DM,DO,DZ,EC,EE,EG,ES,FI,GB,GD,GE,GH,GM,GT,HN,HR,HU,ID,IL,IN,IR,IS,JO,JP,KE,KG,KH,KN,KP,KR,KW,KZ,LA,LC,LK,LR,LS,LU,LY,MA,MD,ME,MG,MK,MN,MW,MX,MY,MZ,NA,NG,NI,NO,NZ,OM,PA,PE,PG,PH,PL,PT,QA,RO,RS,RU,RW,SA,SC,SD,SE,SG,SK,SL,ST,SV,SY,TH,TJ,TM,TN,TR,TT,TZ,UA,UG,US,UZ,VC,VN,WS,ZA,ZM,ZW

(72)発明者 サストリー，アケッラ
アメリカ合衆国、９５１２４ カリフォルニア州、サン・ノゼ、ロジック・ドライブ、２１００
(72)発明者 スレンダラン，リシ
アメリカ合衆国、９５１２４ カリフォルニア州、サン・ノゼ、ロジック・ドライブ、２１００
(72)発明者 ジェイムズ－ロックスビー，フィリップ・ビィ
アメリカ合衆国、９５１２４ カリフォルニア州、サン・ノゼ、ロジック・ドライブ、２１００
(72)発明者 ベイリス，サミュエル・アール
アメリカ合衆国、９５１２４ カリフォルニア州、サン・ノゼ、ロジック・ドライブ、２１００
(72)発明者 カタイル，ビノッド・ケイ
アメリカ合衆国、９５１２４ カリフォルニア州、サン・ノゼ、ロジック・ドライブ、２１００
(72)発明者 アグラワル，アジト・ケイ
アメリカ合衆国、９５１２４ カリフォルニア州、サン・ノゼ、ロジック・ドライブ、２１００
(72)発明者 ウィッティヒ，ラルフ・デー
アメリカ合衆国、９５１２４ カリフォルニア州、サン・ノゼ、ロジック・ドライブ、２１００

Ｆターム（参考） 5B045 BB15 BB37 BB54 GG00 GG18 HH06
5B062 CC04 DD09 JJ01
5B146 AA22 GA01 GC01 GC11