(19)日本国特許庁(JP)　(12)公表特許公報(A)　(11)特許出願公表番号
特表2006-522406
(P2006-522406A)
(43)公表日　平成18年9月28日(2006.9.28)

(51)Int.Cl.	F I	テーマコード（参考）
G06F 15/82 (2006.01)	G06F 15/82 640Z	
	G06F 15/82 630Z	

審査請求 未請求　予備審査請求 未請求　（全 33 頁）

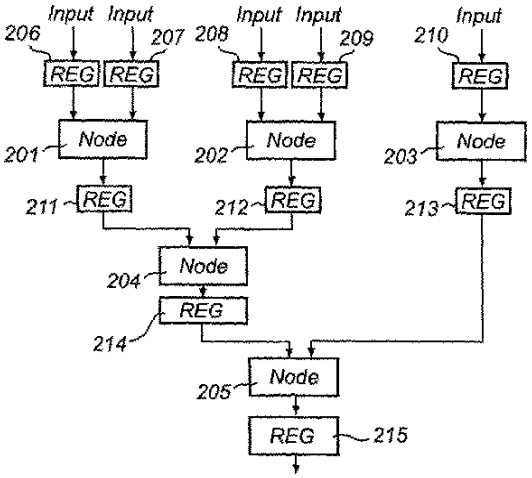
(21)出願番号	特願2006-507967 (P2006-507967)
(86)(22)出願日	平成16年3月17日 (2004.3.17)
(85)翻訳文提出日	平成17年10月13日 (2005.10.13)
(86)国際出願番号	PCT/SE2004/000394
(87)国際公開番号	WO2004/084086
(87)国際公開日	平成16年9月30日 (2004.9.30)
(31)優先権主張番号	0300742-4
(32)優先日	平成15年3月17日 (2003.3.17)
(33)優先権主張国	スウェーデン(SE)

(71)出願人　505351072
マイトリオニクス　エービー
スウェーデン国　エス　－　２２３　７０　ルンド、　オレ　レメルス　ベーグ　１６、イデオン
(74)代理人　100066692
弁理士　浅村　皓
(74)代理人　100072040
弁理士　浅村　肇
(74)代理人　100091339
弁理士　清水　邦明
(74)代理人　100094673
弁理士　林　鈺三

最終頁に続く

(54)【発明の名称】データ・フロー・マシン

(57)【要約】
高レベル原始コード仕様からディジタル論理記述を生成するための方法が開示される。原始コード仕様の少なくとも一部は、少なくとも１つの入力あるいは少なくとも１つの出力を有する機能ノード、および機能ノード間の相互接続を示す接続を含む、多元有向グラフ表現にコンパイルされる。ハードウェア構成要素は、グラフの各機能ノードに対して、そして機能ノード間の各接続に対して定義される。最後に、グラフの機能ノードの各々に対する発火規則が定義される。



【特許請求の範囲】
【請求項１】
少なくとも１つの入力あるいは少なくとも１つの出力を有する機能ノード、および上記機能ノード間の相互接続を示す接続を含むグラフ表現からディジタル論理回路を実装するための、ディジタル制御パラメータを生成する方法であって、
上記グラフの各機能ノードに対するハードウェア構成要素を特定するディジタル制御パラメータを定義することと、
上記機能ノード間の各接続に対するハードウェア構成要素を特定するディジタル制御パラメータを定義することと、
上記グラフの上記機能ノードの各々に対する上記ハードウェア構成要素における発火規則を特定するディジタル制御パラメータを定義すること、
を特徴とする、上記方法。
【請求項２】
請求項１に記載の方法であって、
上記グラフ表現は有向グラフであることを特徴とする、上記方法。
【請求項３】
請求項１あるいは２に記載の方法であって、
上記グラフ表現は、高レベル原始コード仕様から生成されていることを特徴とする、上記方法。
【請求項４】
請求項１から３に記載の方法であって、
上記機能ノード間の少なくとも１つの接続に対して、独立して並行してアクセスされることができるオンチップ記憶素子を特定するディジタル制御パラメータを定義することを特徴とする、上記方法。
【請求項５】
請求項１から４に記載の方法であって、
上記機能ノード間の少なくとも１つの接続に対して、ディジタル・レジスタを特定するディジタル制御パラメータを定義することを特徴とする、上記方法。
【請求項６】
請求項１から４に記載の方法であって、
上記機能ノード間の少なくとも１つの接続に対して、少なくとも１つのフリップフロップを特定するディジタル制御パラメータを定義することを特徴とする、上記方法。
【請求項７】
請求項１から４に記載の方法であって、
上記機能ノード間の少なくとも１つの接続に対して、少なくとも１つのラッチを特定するディジタル制御パラメータを定義することを特徴とする、上記方法。
【請求項８】
先行する請求項のいずれかに記載の方法であって、
各機能ノードに対して、組合せ論理を特定するディジタル制御パラメータを定義することを特徴とする、上記方法。
【請求項９】
請求項１から７に記載の方法であって、
少なくとも１つの機能ノードに対して、少なくとも１つの状態機械を特定するディジタル制御パラメータを定義することを特徴とする、上記方法。
【請求項１０】
少なくとも１つの入力あるいは少なくとも１つの出力を有する機能ノード、および上記機能ノード間の相互接続を示す接続を含むグラフ表現からディジタル論理回路を実装するための、ディジタル制御パラメータを生成するための装置であって、
上記グラフの各機能ノードに対して、ハードウェア構成要素を特定するディジタル制御パラメータを定義し、

上記機能ノード間の各接続に対して、ハードウェア構成要素を特定するディジタル制御パラメータを定義し、
上記グラフの上記機能ノードの各々に対して、上記ハードウェア構成要素における発火規則を特定するディジタル制御パラメータを定義するよう適合されたことを特徴とする、上記装置。
【請求項１１】
請求項１０に記載の装置であって、
上記グラフ表現は有向グラフであることを特徴とする、上記装置。
【請求項１２】
請求項１０あるいは１１に記載の装置であって、
上記グラフ表現は、高レベル原始コード仕様から生成されていることを特徴とする、上記装置。
【請求項１３】
請求項１０から１２に記載の装置であって、
上記機能ノード間の少なくとも１つの接続に対して、独立して並行してアクセスされることができるオンチップ記憶素子を特定するディジタル制御パラメータを定義するよう適合されていることを特徴とする、上記装置。
【請求項１４】
請求項１０から１３に記載の装置であって、
上記機能ノード間の少なくとも１つの接続に対して、ディジタル・レジスタを特定するディジタル制御パラメータを定義するよう適合されていることを特徴とする、上記装置。
【請求項１５】
請求項１０から１３に記載の装置であって、
上記機能ノード間の少なくとも１つの接続に対して、少なくとも１つのフリップフロップを特定するディジタル制御パラメータを定義するよう適合されていることを特徴とする、上記装置。
【請求項１６】
請求項１０から１３に記載の装置であって、
上記機能ノード間の少なくとも１つの接続に対して、少なくとも１つのラッチを特定するディジタル制御パラメータを定義するよう適合されていることを特徴とする、上記装置。
【請求項１７】
請求項１０から１６に記載の装置であって、
各機能ノードに対して、組合せ論理を特定するディジタル制御パラメータを定義するよう適合されていることを特徴とする、上記装置。
【請求項１８】
請求項１０から１６に記載の装置であって、
各機能ノードに対して、少なくとも１つの状態機械を特定するディジタル制御パラメータを定義するよう適合されていることを特徴とする、上記装置。
【請求項１９】
データ・フロー・マシンであって、
データ変換を実行する第１の組のハードウェア構成要素と、
上記第１の組のハードウェア構成要素を相互に接続する第２の組のハードウェア構成要素と、
上記第１の組のハードウェア構成要素の各々に対して、少なくとも１つの発火規則を確定する電子回路と、
を特徴とする、上記データ・フロー・マシン。
【請求項２０】
請求項１９に記載のデータ・フロー・マシンであって、
上記グラフ表現は有向グラフであることを特徴とする、上記データ・フロー・マシン。

【請求項２１】
請求項１９あるいは２０に記載のデータ・フロー・マシンであって、
上記第２の組のハードウェア構成要素の少なくとも１つの構成要素は、独立して並行してアクセスされることができるオンチップ記憶素子の形式であることを特徴とする、上記データ・フロー・マシン。
【請求項２２】
請求項１９から２１に記載のデータ・フロー・マシンであって、
上記第２の組のハードウェア構成要素の少なくとも１つの構成要素は、レジスタの形式であることを特徴とする、上記データ・フロー・マシン。
【請求項２３】
請求項１９から２１に記載のデータ・フロー・マシンであって、
上記第２の組のハードウェア構成要素の少なくとも１つの構成要素は、フリップフロップの形式であることを特徴とする、上記データ・フロー・マシン。
【請求項２４】
請求項１９から２１に記載のデータ・フロー・マシンであって、
上記第２の組のハードウェア構成要素の少なくとも１つの構成要素は、ラッチの形式であることを特徴とする、上記データ・フロー・マシン。
【請求項２５】
請求項１９から２４に記載のデータ・フロー・マシンであって、
上記第１の組のハードウェア構成要素は、組合せ論理の形式であることを特徴とする、上記データ・フロー・マシン。
【請求項２６】
請求項１９から２４に記載のデータ・フロー・マシンであって、
上記第１の組のハードウェア構成要素の少なくとも１つの構成要素は、少なくとも１つの状態機械の形式であることを特徴とする、上記データ・フロー・マシン。
【請求項２７】
請求項１９から２６に記載のデータ・フロー・マシンであって、
上記データ・フロー・マシンは、ＡＳＩＣ、ＦＰＧＡ、ＣＰＬＤあるいは他のＰＬＤによって実現されていることを特徴とする、上記データ・フロー・マシン。
【請求項２８】
ディジタル・コンピュータ機能を有する電子装置のメモリ内に直接ロード可能なコンピュータ・プログラム製品であって、上記製品が上記電子装置によって実行されるときに、請求項１から９のいずれかの上記ステップを実行するためのソフトウェア・コード部分を含む、上記コンピュータ・プログラム製品。
【請求項２９】
請求項２８に記載のコンピュータ・プログラム製品であって、コンピュータ読取り可能媒体上に組み込まれていることを特徴とする、上記コンピュータ・プログラム製品。
【請求項３０】
機能ノードを含むグラフ表現からディジタル論理回路を実現するためのディジタル制御パラメータを生成する方法であって、上記機能ノードは、少なくとも１つの入力あるいは少なくとも１つの出力、および上記機能ノード間の相互接続を示す接続を含み、
少なくとも第１および第２の機能ノードに対して、併合されたハードウェア構成要素を特定するディジタル制御パラメータを定義することと、上記ディジタル制御パラメータは、上記第１および第２の機能ノード双方の機能を実行するために上記併合されたハードウェア構成要素を特定し、
上記第１および第２の機能ノードの上記併合から生じる上記ハードウェア構成要素に対して発火規則を特定するディジタル制御パラメータを定義すること、
を含む、上記方法。
【請求項３１】
請求項３０に記載の方法において、上記生成されたディジタル制御パラメータは、上記

第１の機能ノードの上記少なくとも１つの出力と、上記第２の機能ノードの上記少なくとも１つの入力との間の直接接続を特定することを特徴とする、上記方法。
【請求項３２】
請求項３０あるいは３１のいずれかに記載の方法であって、
上記第１および第２の機能ノードの上記発火規則とは異なる、上記併合されたハードウェア構成要素に対する発火規則を特定するディジタル制御パラメータを定義することを特徴とする、上記方法。
【請求項３３】
機能ノードを含むグラフ表現からディジタル論理回路を実装するためのディジタル制御パラメータを生成するための装置であって、上記機能ノードは、少なくとも１つの入力あるいは少なくとも１つの出力、および上記機能ノード間の相互接続を示す接続を含み、
少なくとも第１および第２の機能ノードに対して、併合されたハードウェア構成要素を特定するディジタル制御パラメータを定義し、上記ディジタル制御パラメータは、上記第１および第２の機能ノード双方の機能を実行するために上記併合されたハードウェア構成要素を特定し、
上記第１および第２の機能ノードの上記併合から生じる上記ハードウェア構成要素に対して発火規則を特定するディジタル制御パラメータを定義するよう適合されている、上記装置。
【請求項３４】
請求項３３に記載の装置であって、上記第１の機能ノードの上記少なくとも１つの出力と、上記第２の機能ノードの上記少なくとも１つの入力との間の直接接続を特定するディジタル制御パラメータを定義するよう適合されていることを特徴とする、上記装置。
【請求項３５】
請求項３３あるいは３４のいずれかに記載の装置であって、上記第１および第２の機能ノードの上記発火規則とは異なる、上記併合されたハードウェア構成要素における発火規則を特定するディジタル制御パラメータを定義するよう適合されていることを特徴とする、上記装置。
【請求項３６】
ディジタル・コンピュータ機能を有する電子装置のメモリ内に直接ロード可能なコンピュータ・プログラム製品であって、上記製品が上記電子装置によって実行されるときに、請求項３０から３２のいずれかの上記ステップを実行するためのソフトウェア・コード部分を含む、上記コンピュータ・プログラム製品。
【請求項３７】
請求項３６に記載のコンピュータ・プログラム製品であって、コンピュータ読取り可能媒体上に組み込まれていることを特徴とする、上記コンピュータ・プログラム製品。
【請求項３８】
データ・フロー・マシンにおける第１および第２の相互接続ハードウェア構成要素を起動する方法であって、
上記第１のハードウェア構成要素に第１のディジタル・データ要素を提供し、上記第１のハードウェア構成要素に第１のディジタル・データ要素を提供することは、上記第１のハードウェア構成要素を起動することと、
上記第１のディジタル・データ要素を、上記第１のハードウェア構成要素から上記第２のハードウェア構成要素に転送し、上記第１のディジタル・データ要素を上記第２のハードウェア構成要素において受け取ることは、上記第２のハードウェア構成要素を起動し、上記第１のディジタル・データ要素を上記第１のハードウェア構成要素から引き渡すことは、上記第１のハードウェア構成要素を不能状態にすることと、
を含む、上記方法。
【請求項３９】
請求項３８に記載の方法において、上記第１のハードウェア構成要素は、上記第１のディジタル・データ要素の上記引渡しの後で、第２のディジタル・データ要素を提供されて

もよいことを特徴とする、上記方法。
【請求項４０】
請求項３８あるいは３９のいずれかに記載の方法において、上記ディジタル・データ要素は、上記第１のハードウェア構成要素において生成されることを特徴とする、上記方法。
【請求項４１】
請求項３８あるいは３９のいずれかに記載の方法において、上記ディジタル・データ要素は、個別のハードウェア構成要素において生成され、上記第１のハードウェア構成要素に転送されることを特徴とする、上記方法。
【請求項４２】
請求項３８から４１のいずれかに記載の方法において、上記ディジタル・データ要素は、上記第２のハードウェア構成要素から引き渡され、上記第１のハードウェア構成要素に戻されることを特徴とする、上記方法。
【請求項４３】
第１および第２の相互接続ハードウェア構成要素を含むデータ・フロー・マシンであって、
上記第１のハードウェア構成要素は、第１のディジタル・データ要素を提供され、上記第１のディジタル・データ要素が上記第１のハードウェア構成要素内に存在する限り、起動されるよう適合されることと、
上記第１のハードウェア構成要素は、上記第１のディジタル・データ要素を、上記第１のハードウェア構成要素から上記第２のハードウェア構成要素に転送するよう適合されることとを特徴とし、
上記第２のハードウェア構成要素は、少なくとも上記第１のディジタル・データ要素の受け取りの結果として起動されるよう適合され、上記第１のハードウェア構成要素は、上記ディジタル・データ要素の上記引渡しの結果として、不能状態になるよう適合されている、
上記データ・フロー・マシン。
【請求項４４】
請求項４３に記載のデータ・フロー・マシンにおいて、上記第１のハードウェア構成要素は、上記第１のディジタル・データ要素が上記第２のハードウェア構成要素へ転送された後に、第２のディジタル・データ要素を提供されるよう適合された、上記データ・フロー・マシン。
【請求項４５】
請求項４３あるいは４４のいずれかに記載のデータ・フロー・マシンにおいて、上記第１のハードウェア構成要素は、上記ディジタル・データ要素を生成するよう適合された、上記データ・フロー・マシン。
【請求項４６】
請求項４３あるいは４４のいずれかに記載のデータ・フロー・マシンにおいて、上記第１のハードウェア構成要素は、個別のハードウェア構成要素から上記ディジタル・データ要素を受信するよう適合された、上記データ・フロー・マシン。
【請求項４７】
請求項４３から４６に記載のデータ・フロー・マシンであって、
上記データ・フロー・マシンは、ＡＳＩＣ、ＦＰＧＡ、ＣＰＬＤあるいは他のＰＬＤによって実装されていることを特徴とする、上記データ・フロー・マシン。
【請求項４８】
データ・フロー・マシンにおけるデータの完全性を保証する方法であって、少なくとも１つの機能停止線が、上記データ・フロー・マシン内にデータ経路が提供されるように配置された少なくとも第１および第２のハードウェア構成要素に接続し、データのフローを中断するために提供され、該データのフローは、処理サイクルの間、データ経路において上記第１のハードウェア構成要素から上記第２のハードウェア構成要素に進み、機能停止

信号が上記機能停止線上で活動状態である場合、
上記機能停止信号を、上記第２のハードウェア構成要素から、第１のオンチップ記憶素子の第１の入力において受信することと、
データを、上記第１のハードウェア構成要素から、第２のオンチップ記憶素子の第１の入力において受信することと、
上記第１および第２のオンチップ記憶素子における上記受信データおよび上記受信機能停止信号を、少なくとも１処理サイクルの間バッファに入れることと、
上記バッファに入れられた機能停止信号を、上記第１のハードウェア構成要素に、上記第１のオンチップ記憶素子の第１の出力において提供することと、
上記バッファに入れられたデータを、上記第２のハードウェア構成要素に、上記第２のオンチップ記憶素子の第１の出力において提供することと、
を特徴とする、上記方法。
【請求項４９】
請求項４８に記載の方法において、少なくとも１つのオンチップ記憶素子がレジスタである、上記方法。
【請求項５０】
データ・フロー・マシンにおけるデータの完全性を保証するための装置であって、少なくとも１つの機能停止線が、上記データ・フロー・マシン内にデータ経路が提供されるように配置された少なくとも第１および第２のハードウェア構成要素に接続し、データのフローを中断するために提供され、該データのフローは、処理サイクルの間、データ経路において上記第１のハードウェア構成要素から上記第２のハードウェア構成要素に進み、機能停止信号が上記機能停止線上で活動状態である場合、
上記機能停止信号を、上記第２のハードウェア構成要素から、第１のオンチップ記憶素子の第１の入力において受信し、
データを、上記第１のハードウェア構成要素から、第２のオンチップ記憶素子の第１の入力において受信し、
上記第１および第２のオンチップ記憶素子における上記受信データおよび上記受信機能停止信号を、少なくとも１処理サイクルの間バッファに入れ、
上記バッファに入れられた機能停止信号を、上記第１のハードウェア構成要素に、上記第１のオンチップ記憶素子の第１の出力において提供し、
上記バッファに入れられたデータを、上記第２のハードウェア構成要素に、上記第２のオンチップ記憶素子の第１の出力において提供する、
よう適合されていることを特徴とする、上記装置。
【請求項５１】
請求項４８に記載の方法において、少なくとも１つのオンチップ記憶素子がレジスタである、上記方法。
【請求項５２】
少なくとも１つの入力あるいは少なくとも１つの出力を有する機能ノード、および上記機能ノード間の相互接続を示す接続を含むグラフ表現からディジタル論理回路を実装するための、ディジタル制御パラメータを生成する方法であって、
上記機能ノードあるいは上記機能ノード間の上記接続に対して、少なくとも第１の組のハードウェア構成要素を特定するディジタル制御パラメータを定義することと、
データ要素が、それらが上記第１の組のハードウェア構成要素に入るのと同じ順序で上記第１の組のハードウェア構成要素から送り出されるように、少なくとも１つの第１の組のハードウェア構成要素からデータ要素が送り出される順序を決める、少なくとも１つの再順序付けハードウェア構成要素を特定するディジタル制御パラメータを定義することと、
を特徴とする、上記方法。
【請求項５３】
請求項５２に記載の方法であって、

上記グラフ表現は有向グラフであることを特徴とする、上記方法。
【請求項５４】
請求項５２あるいは５３のいずれかに記載の方法であって、
上記グラフ表現は、高レベル原始コード仕様から生成されていることを特徴とする、上記方法。
【請求項５５】
請求項５２から５４のいずれかに記載の方法であって、
上記機能ノード間の少なくとも１つの接続に対して、独立して並行してアクセスされることができるオンチップ記憶素子を特定するディジタル制御パラメータを定義することを特徴とする、上記方法。
【請求項５６】
請求項５２から５５のいずれかに記載の方法であって、
上記機能ノード間の少なくとも１つの接続に対して、ディジタル・レジスタを特定するディジタル制御パラメータを定義することを特徴とする、上記方法。
【請求項５７】
請求項５２から５５のいずれかに記載の方法であって、
上記機能ノード間の少なくとも１つの接続に対して、少なくとも１つのフリップフロップを特定するディジタル制御パラメータを定義することを特徴とする、上記方法。
【請求項５８】
請求項５２から５５のいずれかに記載の方法であって、
上記機能ノード間の少なくとも１つの接続に対して、少なくとも１つのラッチを特定するディジタル制御パラメータを定義することを特徴とする、上記方法。
【請求項５９】
少なくとも１つの入力あるいは少なくとも１つの出力を有する機能ノード、および上記機能ノード間の相互接続を示す接続を含むグラフ表現からディジタル論理回路を実装するための、ディジタル制御パラメータを生成するための装置であって、
上記機能ノードあるいは上記機能ノード間の上記接続に対して、少なくとも第１の組のハードウェア構成要素を特定するディジタル制御パラメータを定義し、
データ要素が、それらが上記第１の組のハードウェア構成要素に入るのと同じ順序で上記第１の組のハードウェア構成要素から送り出されるように、少なくとも１つの第１の組のハードウェア構成要素からデータ要素が送り出される順序を決める、少なくとも１つの再順序付けハードウェア構成要素を特定するディジタル制御パラメータを定義する、
よう適合されていることを特徴とする、上記装置。
【請求項６０】
請求項５９に記載の装置であって、
上記グラフ表現は有向グラフであることを特徴とする、上記装置。
【請求項６１】
請求項５９あるいは６０に記載の装置であって、
上記グラフ表現は、高レベル原始コード仕様から生成されていることを特徴とする、上記装置。
【請求項６２】
請求項５９から６１に記載の装置であって、
上記機能ノード間の少なくとも１つの接続に対して、独立して並行してアクセスされることができるオンチップ記憶素子を特定するディジタル制御パラメータを定義するよう適合されていることを特徴とする、上記装置。
【請求項６３】
請求項５９から６２に記載の装置であって、
上記機能ノード間の少なくとも１つの接続に対して、ディジタル・レジスタを特定するディジタル制御パラメータを定義するよう適合されていることを特徴とする、上記装置。
【請求項６４】

請求項５９から６２に記載の装置であって、
上記機能ノード間の少なくとも１つの接続に対して、少なくとも１つのフリップフロップを特定するディジタル制御パラメータを定義するよう適合されていることを特徴とする、上記装置。
【請求項６５】
請求項５９から６２に記載の装置であって、
上記機能ノード間の少なくとも１つの接続に対して、少なくとも１つのラッチを特定するディジタル制御パラメータを定義するよう適合されていることを特徴とする、上記装置。
【請求項６６】
データ・フロー・マシンであって、
データ変換を実行する第１の組のハードウェア構成要素と、
少なくとも１つの第１の組のハードウェア構成要素から送り出されるデータ要素を順序付けし、データ要素が上記第１の組のハードウェア構成要素から、それらが上記第１の組のハードウェア構成要素に入ったのと同じ順序で送り出されるようにする、少なくとも１つの再順序付けハードウェア構成要素、
を含む、上記データ・フロー・マシン。
【請求項６７】
請求項６６に記載のデータ・フロー・マシンであって、
上記グラフ表現は有向グラフであることを特徴とする、上記データ・フロー・マシン。
【請求項６８】
請求項６６あるいは６７に記載のデータ・フロー・マシンであって、
上記第２および第３の組のハードウェア構成要素の少なくとも１つの構成要素は、独立して並行してアクセスされることができるオンチップ記憶素子の形式であることを特徴とする、上記データ・フロー・マシン。
【請求項６９】
請求項６６から６８に記載のデータ・フロー・マシンであって、
上記第２および第３の組のハードウェア構成要素の少なくとも１つの構成要素は、レジスタの形式であることを特徴とする、上記データ・フロー・マシン。
【請求項７０】
請求項６６から６８に記載のデータ・フロー・マシンであって、
上記第２および第３の組のハードウェア構成要素の少なくとも１つの構成要素は、フリップフロップの形式であることを特徴とする、上記データ・フロー・マシン。
【請求項７１】
請求項６６から６８に記載のデータ・フロー・マシンであって、
上記第２および第３の組のハードウェア構成要素の少なくとも１つの構成要素は、ラッチの形式であることを特徴とする、上記データ・フロー・マシン。
【請求項７２】
請求項６６から７１に記載のデータ・フロー・マシンであって、
上記データ・フロー・マシンは、ＡＳＩＣ、ＦＰＧＡ、ＣＰＬＤあるいは他のＰＬＤによって実装されていることを特徴とする、上記データ・フロー・マシン。
【請求項７３】
ディジタル・コンピュータ機能を有する電子装置のメモリ内に直接ロード可能なコンピュータ・プログラム製品であって、上記製品が上記電子装置によって実行されるときに、請求項５２から５８のいずれかの上記ステップを実行するためのソフトウェア・コード部分を含む、上記コンピュータ・プログラム製品。
【請求項７４】
請求項７３に記載のコンピュータ・プログラム製品であって、コンピュータ読取り可能媒体上に組み込まれている、上記コンピュータ・プログラム製品。
【請求項７５】

請求項３０から３２のいずれかに記載の方法において、上記グラフ表現は、高レベル原始コード仕様から生成されている、上記方法。
【請求項７６】
請求項３３から３５のいずれかに記載の装置において、上記グラフ表現は、高レベル原始コード仕様から生成されている、上記装置。
【請求項７７】
請求項４３から４６に記載のデータ・フロー・マシンにおいて、上記ディジタル・データ要素は、上記第２のハードウェア構成要素から引き渡され、上記第１のハードウェア構成要素に戻される、上記データ・フロー・マシン。
【発明の詳細な説明】
【技術分野】
【０００１】
（技術分野）
一般に、本発明は、データ処理方法および装置に関し、特に、細粒度並列処理および大規模パイプライン深度を使用するデータ・フロー・マシンにより、ディジタル・ハードウェアにおける高速データ処理を実行するための方法および装置に関するものである。
【背景技術】
【０００２】
ディジタル回路を設計するための迅速で容易な方法を提供するために、近年、ハードウェア記述のための使いやすいプログラミング言語に対する多くの異なる方法が使用されている。データ・フロー・マシンをプログラムする場合、ハードウェア記述言語とは異なる言語が使用されるかもしれない。原則として、データ・フロー・マシン上の特定のタスクを実行するためのアルゴリズム記述は、記述そのものを含みさえすればよいが、集積回路において直接実行されるべきアルゴリズム記述は、ハードウェアにおけるアルゴリズムの特定の実現の多くの詳細を含まなければならない。例えば、ハードウェア記述は、最適なクロック周波数を提供するためのレジスタの配置や、どの乗算器を使用するか、等に関する情報を含まなければならない。
【０００３】
長年の間、データ・フロー・マシンは並列計算処理に対する優秀なモデルとして考えられてきており、従って、効率的なデータ・フロー・マシンを設計するための多くの試みが行われてきた。様々な理由により、データ・フロー・マシンを設計しようとする初期の試みは、他の使用可能な並列計算処理技術と比較して、計算性能に関して悪い結果を出してきた。
【０００４】
プログラム原始コードを翻訳する場合、今日使用可能なほとんどのコンパイラは、コンパイルされたプログラムの性能を最適化するために、データ・フロー解析およびデータ・フロー記述（データ・フロー・グラフ、あるいはＤＦＧとして知られている）を使用している。アルゴリズム上で実行されたデータ・フロー解析は、データ・フロー・グラフを生成する。データ・フロー・グラフは、アルゴリズム内に存在するデータ従属性を示す。より具体的には、データ・フロー・グラフは、通常、処理されているデータ上でアルゴリズムが実行する特定の動作を示すノード、およびグラフにおけるノード間の相互接続を示すアーク、を含む。データ・フロー・グラフは従って、特定のアルゴリズムの要約記述であり、アルゴリズムを解析するために使用される。他方、データ・フロー・マシンは、データ・フロー・グラフに基づき、実際にアルゴリズムを実行することができる計算機械である。
【０００５】
データ・フロー・マシンは、ノイマン型アーキテクチャ（パーソナル・コンピュータにおける通常のプロセッサはノイマン型アーキテクチャの１例である）のような制御フロー装置と比較すると、非常に異なる方法で動作する。データ・フロー・マシンにおいては、プログラムは、プロセッサによって実行されるべき一連の動作ではなく、データ・フロー

・グラフである。データは、データ・フロー・グラフのアーク上に存在するトークンとして知られるパケットに編成される。トークンは、ビット、浮動小数点数、配列、等のような、アークによって接続されるノードによって動作されるべき任意のデータ構造を含むことができる。データ・フロー・マシンの型によって、各アークは、最大限、単一のトークン（静的データ・フロー・マシン）、固定数のトークン（同期データ・フロー・マシン）、あるいは不特定数のトークン（動的データ・フロー・マシン）、を保持することができる。

【０００６】

データ・フロー・マシンにおけるノードは、それらの動作を実行することができるように、トークンが十分な数の入力アーク上に現れるのを待ち、トークンが現れるとこれらのトークンを消費し、それらの出力アーク上に新しいトークンを生成する。例えば、２つのトークンの加算を実行するノードは、その双方の入力上にトークンが現れるまで待ち、これらの２つのトークンを消費し、それから結果（この場合、入力トークンのデータの合計）を、その出力アーク上に新しいトークンとして生成する。

【０００７】

ＣＰＵにおいて実行されるように、条件付き分岐によりデータ上で実行する異なる動作を選択するのではなく、データ・フロー・マシンは、条件付き分岐によりデータを異なるノードに方向付ける。従って、データ・フロー・マシンは、特定の出力上に選択的にトークンを生成することができるノード（スイッチ・ノードと呼ばれる）、および、特定の入力上のトークンを選択的に消費することができるノード（併合ノードと呼ばれる）を有する。一般的なデータ・フロー操作ノードの他の例は、データ・フローから選択的にトークンを除去するゲート・ノードである。多くの他のデータ・フロー操作ノードもまた、可能である。

【０００８】

グラフにおける各ノードは、グラフにおける全ての他のノードから独立してその動作を潜在的に実行することができる。ノードがその適切な入力アーク上にデータを有するとすぐ、そして、その適切な出力アーク上に結果を生成する空間があれば、ノードはその動作を実行することができる（発火として知られている）。ノードは、他のノードが発火できるかどうかに関わらず、発火する。従って、制御フロー装置におけるような、ノードの動作が実行される特定の順序はない。データ・フロー・グラフにおける動作の実行の順序は、関連性がない。実行の順序は、例えば、発火することができる全てのノードの同時実行であることもできる。

【０００９】

上記のように、データ・フロー・マシンは、その設計により、通常３つの異なる分類に分けられる。静的データ・フロー・マシン、動的データ・フロー・マシン、および同期データ・フロー・マシン。

【００１０】

静的データ・フロー・マシンにおいて、対応するデータ・フロー・グラフにおける全てのアークは、全ての時点において単一のトークンのみを保持することができる。

【００１１】

動的データ・フロー・マシンにおいて、各アークは、受信ノードがトークンを受信する準備ができるのを待つ間、不特定数のトークンを保持することができる。このことにより、データ・フロー・マシンを設計する時には再帰深度が不明である再帰的手続きの構築が可能になる。このような手続きは、再帰において処理されているデータを逆にするかもしれない。このことは、再帰が終了した後に計算を実行する時、トークンのマッチングが間違っているという結果になるかもしれない。

【００１２】

上記の状況は、プロトコルにおける全てのトークンの通し番号を示すマーカを追加することで処理することができる。再帰の内側のトークンの通し番号は、連続的に監視され、トークンが再帰を抜ける時、再帰の外側のトークンとマッチしない限り、進むことができ

ない。

【００１３】

再帰が末端再帰でない場合、再帰が通常の（ノイマン型の）プロセッサの使用により実行される場合にコンテキストがスタック上に記憶されるのと同様に、コンテキストは全ての再帰呼出しにおいてバッファに格納されなければならない。最後に、動的データ・フロー・マシンは、データ依存再帰を並行して実行することができる。

【００１４】

同期データ・フロー・マシンは、受信ノードが自身を準備している間トークンをアーク上で待機させる能力がなくても、動作することができる。代わりに、各ノードに対するトークンの生成と消費との間の関係は、前もって計算される。この情報をもって、どのようにノードを配置するかを決定し、同時にアーク上に存在するかもしれないトークンの数に関してアークに大きさを割り当てることができる。従って、各ノードが後続のノードが消費するのと同じ数のトークンを生成することを保証することができる。システムはそれから、後続のノードが常にデータを消費するので、全てのノードが常にデータを生成することができるよう設計することができる。欠点は、データ依存再帰のようなどんな不確定の遅延も、構成において存在することができないことである。

【００１５】

データ・フロー・マシンは、最も一般的には、従来のＣＰＵにおいて実行されるコンピュータ・プログラムによって実施される。しばしばコンピュータのクラスタが使用され、あるいは何らかのプリント回路板上のＣＰＵの配列が使用される。データ・フロー・マシンを使用する主な目的は、それらの並列処理を利用して実験的なスーパコンピュータを構築することである。データ・フロー・マシンをハードウェアに直接構築する多くの試みがなされてきた。このことは、特定用途向け集積回路（ＡＳＩＣ）においていくつかのプロセッサを作成することによって行われてきた。回路板上のプロセッサを使用することに対するこの方法の主な利点は、同じＡＳＩＣ上のプロセッサ間のより速い通信速度である。今日まで、計算処理のためにデータ・フロー・マシンを使用するどの試みも、商業的に成功していない。

【００１６】

書替え可能ゲートアレイ（ＦＰＧＡ）および他のプログラム可能論理デバイス（ＰＬＤ）もまた、ハードウェア構築のために使用することができる。ＦＰＧＡは、急いで再構成可能なシリコン・チップである。これらは、小さなランダム・アクセス・メモリ、通常、スタティック・ランダム・アクセス・メモリ（ＳＲＡＭ）のアレイに基づいている。各ＳＲＡＭは、ブール関数のためのルックアップ・テーブルを保持し、従ってＦＰＧＡがどんな論理演算も実行することが可能となる。ＦＰＧＡはまた、同様に構成可能な経路指定資ソースを保持し、信号がＳＲＡＭからＳＲＡＭへと渡ることを可能にしている。

【００１７】

シリコン・チップの論理演算をＳＲＡＭに割り当て、経路指定資ソースを構成することによって、ＦＰＧＡ表面上に適合するのに十分小さい任意のハードウェア構成を実現することができる。ＦＰＧＡは、ＡＳＩＣと比較して、同じ広さのシリコン表面上に、かなりより少ない論理演算を実現することができる。ＦＰＧＡの利点は、単にＳＲＡＭルックアップ・テーブルに新しい値を入れ、経路指定を変更することによって、任意の他のハードウェア構成に変更することができることである。ＦＰＧＡは、任意のハードウェア構成を受け入れることができ、任意の他のハードウェア構成に非常に短時間（１００ミリ秒より短い時間）で変更することができる、空のシリコン表面として見なすことができる。

【００１８】

他の一般的なＰＬＤは、ヒューズ・リンクされていてもよく、従って永久的に構成されている。ヒューズ・リンクされたＰＬＤのＡＳＩＣに対する主な利点は、構成が容易なことである。ＡＳＩＣを製造するためには、非常に高価で複雑な処理が要求される。対照的にＰＬＤは、簡単なツールによって数分で構成することができる。ヒューズ・リンクされたＰＬＤおよびＦＰＧＡの双方に対して、不利益のいくつかを克服することができる、Ｐ

ＬＤのためのいくつかの発展した技術がある。

【００１９】

一般に、ＦＰＧＡをプログラムするためには、ＦＰＧＡの販売者によって提供される位置および経路ツールを使用しなければならない。位置および経路ソフトウェアは通常、合成ソフトウェアからのネットリストか、あるいはハードウェア記述言語が直接合成するハードウェア記述言語（ＨＤＬ）からの原始コードを受け取る。位置および経路ソフトウェアはそれから、プログラミング装置においてＦＰＧＡをプログラムするために使用される記述ファイルに、ディジタル制御パラメータを出力する。同様の技術が、他のＰＬＤに対して使用されている。

【００２０】

集積回路を設計する場合、回路を状態機械として設計することが一般的な実現である。状態機械はハードウェアの構築を簡素化するフレームワークを提供するからである。状態機械は、データが先の計算によって多様なパターンで論理演算を通って流れる、データの複雑なフローを実現する場合に特に有用である。

【００２１】

状態機械はまた、ハードウェア構成要素の再使用を可能にし、従って回路の物理的な大きさを最適化する。このことにより、集積回路をより低いコストで製造することが可能になる。

【００２２】

専用ハードウェアを使用するデータ・フロー・マシンの従来の構築は、状態機械あるいは専用ＣＰＵ（状態機械の特別な場合である）を互いに接続することに基づいている。これらはしばしば専用経路指定論理と、そしてある場合には専用メモリと接続される。より具体的には、データ・フロー・マシンの初期の設計において、状態機械は、データ・フロー・マシンの動作をエミュレートするために使用されている。さらに、初期のデータ・フロー・マシンはまた一般に、動的データ・フロー・マシンの形式であり、特殊トークン・マッチングおよび再順序付け構成要素がしばしば使用されている。

【００２３】

米国特許第５，０２１，９４７号は、上記による多重処理システムを開示し、このシステムは、３次元構造において５１２個までの要素プロセッサ（ＰＥ）を配置することによって、データ・フロー・マシンをエミュレートする。各ＰＥは、プログラムおよびデータの記憶のためのそれ自身のローカル・メモリを有する完全なＶＬＳＩ実現コンピュータを構成する。データは、処理されるべきデータ、同様に、宛先ＰＥを識別するアドレスおよびＰＥ内のアクターを識別するアドレスの双方を含むデータ・パケットの形式において、異なるＰＥ間で転送される。さらに、ＰＥを相互に接続する通信ネットワークが、信頼性が高いように設計され、誤転送されたメッセージに対する自動再試行、分散バス裁定、代替経路パケット経路指定、その他を有する。加えて、コンピュータのモジュラー性質により、スループットおよび信頼性要求の範囲を満足させるように、より多くの要素プロセッサを加えることができる。

【００２４】

従って、米国特許第５，０２１，９４７号によるエミュレートされたデータ・フロー・マシンの構造は非常に複雑であり、既にデータ・フロー・グラフに表されたデータ・フロー構造を十分に利用していない。また、マシン内を前や後ろに転送されているパケットの監視は、より余分な論理回路の追加を意味する。

【００２５】

ヴェルドチアその他による文献「同質のデータ・フロー・グラフ・モデルを使用する条件付および反復構造」もまた、同質のデータのフローを獲得するために配置された１組のプロセッサを含むデータ・フロー・マシンを開示している。データ・フロー・マシンは、「アルファ」と呼ばれる装置において実現され、「アルファ」は、同じ著者による文献「静的データ・フロー・モデルのためのアクター・ハードウェア設計」に開示されている。しかし、これらの文献によって開示されるマシンは、初期に確立されたデータ・フロー・

グラフの構造に関して最適化されていない。つまり、多くのステップはデータ・フロー・グラフを確立した後に実行されるが、これらのステップは、コンピュータの形式におけるハードウェア装置の使用により、マシンを実現に適するようにするものである。従って、これらの文献により開示されるマシンは、１組の同一のハードウェア装置（コンピュータ）を通る同質のデータのフローを容易にするが、計算効率に関して最適な方法において、ハードウェアにおいてどのようにデータ・フロー・グラフを実装するかは開示していない。

【００２６】

データ・フロー・マシンの形式において多数のプロセッサを有するスーパコンピュータを構築することによって、高度な並列処理を達成することができると期待されている。プロセッサが、多数のＣＰＵあるいは多数のＡＳＩＣから成り、各々が多くの状態機械を含むようにする試みがなされてきた。初期のデータ・フロー・マシンの設計は、ＡＳＩＣにおける（通常プロセッサの形式における）状態機械の使用を含んでいたので、ＦＰＧＡのようなプログラム可能論理デバイスにおいてデータ・フロー・マシンを実現する最も簡単な方法もまた、状態機械を使用することであった。全ての既知のデータ・フロー・マシンに対する一般的な特徴は、確立されたデータ・フロー・グラフのノードは、最終的なハードウェア実装における特定のハードウェア装置（一般に機能単位、ＦＵとして知られている）に対応しないことである。代わりに、特定の時点において偶然使用可能となったハードウェア装置が、データ・フロー・グラフにおいて影響されたノードによって特定された計算を実行するために使用される。データ・フロー・グラフにおける特定のノードが１回以上実行される予定の場合、異なる機能単位が、ノードが実行されるたびに使用されてもよい。

【００２７】

さらに、従来のデータ・フロー・マシンは全て、データ・フロー・マシンの機能を実行するために、状態機械あるいはプロセッサの使用によって実装されてきた。各状態機械は、データ・フロー・グラフにおける任意のノードの機能を実行することができる。このことは、各ノードが任意の機能単位において実行されることができるようにするために必要である。各状態機械は任意のノードの機能を実行することができるので、現在実行しているノード以外の任意の他のノードに対して要求されるハードウェアは休眠状態となる。状態機械（トークン操作のための支援ハードウェアを有することもある）は、データ・フロー・マシンそれ自体の実現であることに留意されたい。データ・フロー・マシンがいくつかの他の手段によって実現され、偶然その機能ノードにおいて状態機械を含むようになったわけではない。

【００２８】

今日使用されているほとんどのプログラミング言語は、いわゆる命令型言語、例えば、ジャバ、フォートラン、およびベーシックのような言語である。これらの言語は、並列処理を緩めることなくデータ・フローとして書き換えることは不可能か、あるいは少なくとも非常に難しい。

【００２９】

代わりに、命令型言語ではなく機能言語を使用すると、データ・フロー・マシンの設計が簡単になる。機能言語は、参照透明性と呼ばれる特徴を呈することに特徴がある。このことは、直接の構成要素表現の意味あるいは値のみが、より大きい複合表現の意味を決定することにおいて重要であることを意味する。表現は、それらが同じ意味を有する場合そしてその場合のみ等しいので、参照透明性は、等しい副表現は、より大きい表現のコンテキストにおいて交換でき、等しい結果を得ることができることを意味する。

【００３０】

動作の実行が、出力データを提供する以外に影響を与える場合（例えば、動作の実行の間のディスプレイ上の読出し）、それは参照透明ではない。動作の実行から生ずる結果が、動作を実行しない場合の結果と同じではないからである。参照透明言語において書かれたプログラムへのあるいはからの全ての通信は、副作用（例えば、メモリ・アクセス、読

出し、その他）と呼ばれる。
【００３１】
国際特許第０１５９５９３号は、ディジタル・ハードウェア実装へのアルゴリズムの高レベルのソフトウェア・ベース記述のコンパイルを開示している。プログラミング言語のセマンティクスは、コンパイル・ツールの使用を通して翻訳され、コンパイル・ツールは、ソフトウェア記述を解析して制御およびデータ・フロー・グラフを生成する。このグラフはここで、最適化、変換、および注釈のために使用される中間形式である。結果としてのグラフはそれから、ハードウェア実装のレジスタ・トランスファ・レベルあるいはネットリスト・レベル記述のいずれかに翻訳される。個別の制御経路が、フロー・グラフにおけるノードがいつデータを隣接するノードに転送するかを決定するために使用される。並列処理は、制御経路およびデータ経路を分けることによって達成することができる。制御経路を使用することにより、「ウェーブフロント処理」を達成することができ、このことは、データが、制御経路によって制御されるウェーブフロントとして、実際のハードウェア実装を通って流れることを意味する。
【００３２】
制御経路の使用は、データ処理を実行している間、ハードウェアの一部のみを使用することができることを意味する。回路の残りの部分は、制御経路が新しいウェーブフロントを送り出すことができるように、最初のウェーブフロントがフロー・グラフを通って経過するのを待っている。
【００３３】
米国特許第６１４５０７３号は、予め設計され検査されたデータ駆動ハードウェア・コアを開示し、このコアは、単一のチップ上に大規模なシステムを作成するために組み立てられる。トークンは、１ビット作動可能信号および１ビット要求信号によって、専用接続を通してコア間を同時に転送される。作動可能－要求信号ハンドシェークは、トークン転送にとって必要十分である。また、接続されたコアの各々は、少なくとも有限状態機械の複雑さでなければならない。最後に、一般的な発火メカニズムの概念はないので、データのフローの条件付再方向付けは実行することができない。従って、どんなデータ・フロー・マシンもこのシステムと共に構築することはできない。むしろ、コア間のデータ交換のためのプロトコルは、最大限コア内においてパイプラインを保持することに焦点を当てている。
【００３４】
ミハイ・ブディウによる文献「特定用途向けハードウェア：ＣＰＵなしでの計算処理」は、汎用計算処理のためのアーキテクチャを開示し、このアーキテクチャは、特定用途向けハードウェアを作成するために再構成可能ハードウェアとコンパイラ技術を組み合わせている。各静的プログラム命令は、専用ハードウェア実装によって表されている。プログラムは、分割位相抽象機械（ＳＡＭ）と呼ばれる小さなフラグメントに分解され、このＳＡＭは、状態機械としてハードウェアに合成され、相互接続ネットワークによって組み合わされる。プログラムの実行の間、ＳＡＭは次の３つの状態のうちの１つであることができる：非活動、活動、あるいは受動。トークンは異なるＳＡＭ間で受け渡され、これらのトークンはＳＡＭが実行を開始できるようにする。このことは、同時に数個のＳＡＭのみが実際のデータ処理を実行できることを意味し、残りのＳＡＭは、トークンが実行を可能にするまで待機している。この手続きにより、低電力消費が達成されるが、同時に、計算処理能力は低下する。
【発明の開示】
【発明が解決しようとする課題】
【００３５】
本発明は、データ処理システムの性能を向上させるための方法を提供しようとするものである。
【００３６】
より具体的には、本発明の目的は、データ・フロー・マシンをハードウェアにおいて実

装し、超並列処理を獲得することによって、システムの計算能力を増加させることである。

【００３７】

本発明の他の目的は、使用可能なハードウェア資源の使用を改善すること、つまり、使用可能な論理回路（ゲート、スイッチ、その他）のより大きい部分を同時に使用することができるようにすることである。

【００３８】

本目的は、高レベル原始コード仕様からディジタル論理の記述を生成するための方法によって達成される。本方法において、原始コード仕様の少なくとも一部は、少なくとも１つの入力あるいは１つの出力を有する機能ノード、および機能ノード間の相互接続を示す接続を含む、多次元有向グラフ表現にコンパイルされる。さらに、ハードウェア構成要素が、グラフの各機能ノードに対して定義され、ここでハードウェア構成要素は、機能ノードによって定義される機能を表す。追加ハードウェア構成要素が、機能ノード間の各接続に対して定義され、ここで追加ハードウェア構成要素は、第１の機能ノードから第２の機能ノードへのデータの転送を表す。最後に、グラフの機能ノードの各々に対する発火規則が定義され、ここで発火規則は、機能ノードがその出力においてデータを提供しその入力においてデータを消費する条件を定義する。

【００３９】

また、本発明の目的は、異なる機能単位間のデータのフローを可能にするために専用制御経路を使用することにより、従来技術のデータ・フロー・マシンにおける計算処理効率における制限を克服することである。加えて、本発明によるハードウェア実装は、外部メモリとの頻繁な通信を必要とすることなく、データ・フロー・マシンにおいて効率的にデータを記憶する結果、従来技術の解決策と比較して計算処理能力を増加させることができる。

【００４０】

本発明は、従って、専用相互接続ＣＰＵあるいは高水準データ交換プロトコルを必要とせずに、効率的な方法で、データ・フロー・グラフによって記述される機能をハードウェアに実装する。本発明は、データ・フロー・マシンとＲＴＬ（レジスタ・トランスファ・レベル）論理との間のセマンティクスにおける類似性を最大限に使用する。つまり、組合せ論理がＣＰＵの代わりに使用され、ハードウェア・レジスタがＲＡＭ（ランダム・アクセス・メモリ）、バックプレーン、あるいはイーサネット（登録商標）・ネットワークの代わりに使用される。

【００４１】

本発明のさらなる目的は、高レベル・プログラミング言語記述からのシリコン・ハードウェアの設計を可能にすることである。高レベル・プログラミング言語とは、特定の型のハードウェアにおけるアルゴリズムの実装ではなく、それ自体におけるアルゴリズムの記述に焦点を当てているプログラミング言語である。高レベル・プログラミング言語と、その言語で書かれたプログラムから自動的に集積回路記述を設計する機能があれば、集積回路の設計のためにソフトウェア工学技術を使用することが可能になる。このことは、低コストであるいはコストをかけずに、多くの異なるハードウェア設計と共に再構成することのできる、ＦＰＧＡおよび他の再構成可能ＰＬＤにとって特に有益である。

【００４２】

多くの異なる、容易に作成できるハードウェア設計から利益を受けるのとは別に、ＦＰＧＡおよび他のＰＬＤは、高レベル言語からのハードウェア記述の自動設計から効率における利益を受けることができる。システムが大容量並列処理を利用することができれば、可能な限り大部分のＰＬＤを有意義な演算で満たすことができ、性能が非常に高くなる。このことは、通常できるだけ小さな設計を作成することに焦点を当てている従来のハードウェア設計とは対照的である。

【００４３】

本発明の他の目的、特徴および利点は、以下の好ましい実施例の詳細な説明からより明

白となるであろう。
【００４４】
本発明の好ましい実施例は、付随する図面を参照しながら以下に説明される。
【課題を解決するための手段】
【００４５】
一般に、原始コード・プログラムのデータ・フロー・グラフへの変換は、データ・フロー解析によって行われる。データ・フロー解析を実行するための単純な方法は次のようなものである。プログラムの全ての出力から開始する。各出力の直接のソースを見つける。それが動作である場合、動作をノードと置き換え、アークと共に出力に加える。ソースが変数である場合、変数をアークと置き換え出力に結合させる。完全に特定された入力を欠く全てのアークおよびノードに対して繰り返す。
【００４６】
図１ａは、それ自体知られているデータ・フロー・グラフを示している。解りやすくするために、本文章において用語ノードは、データ・フロー・グラフにおける機能ノードを示すよう使用される。３つの処理レベルが本図において示されている。最上部ノード１０１、１０２、１０３は、それらの入力において、１つあるいは複数のソースからの入力データを受信し、このデータはグラフを通って流れる間に処理される。最上部ノードによって実行される実際の数学的、論理的あるいは手続き的機能は、データ・フロー・グラフが起ソースとする原始コードに依存するので、各実装に対して特定である。例えば、第１のノード１０１は、２つの入力からのデータの単純な加算を実行してもよく、第２のノード１０２は、第２の入力において受信されたデータからの第１の入力において受信されたデータの減算を実行してもよく、第３のノード１０３は、例えば、その入力において受信されたデータの２による固定乗算を実行してもよい。各ノードに対する入力の数、各ノードにおいて実行される実際の処理、その他は、もちろん、異なる実装に対して異なり、上記の例に制限されない。１つのノードは、例えばより複雑な計算を実行してもよく、あるいは外部メモリにアクセスしさえしてもよい。このことは以下に説明される。
【００４７】
データは、第１のノード・レベルから第２のノード・レベルに流れていて、この場合、ノード１０１および１０２からのデータは、ノード１０１および１０２の出力からノード１０４の入力に転送される。上記の説明に従い、ノード１０４は、その入力において受信した情報に基づき特定のタスクを実行する。
【００４８】
第２のレベルにおける処理の後、データはノード１０４の出力からノード１０５の第１の入力に転送される。ノード１０５は第３のレベルに位置する。図１から解るように、レベル１におけるノード１０３の出力からのデータは、ノード１０５の第２の入力において受信される。ノード１０３と１０５との間には第２のレベルのノードは存在しないという事実は、ノード１０３からのデータは、ノード１０５の第１の入力ノードにおいてデータが使用可能になる前に、ノード１０５の第２の入力において使用可能になることを意味する（各ノードにおいて等しい組合せ遅延があると仮定する）。この状況を効率的に処理するために、各ノードは、ノードがその出力においてデータを提供する条件を定義する、発火規則を備える。
【００４９】
より具体的には、発火規則は、データ・フロー・グラフにおけるデータのフローを制御するメカニズムである。発火規則を使用することにより、データがノードの機能により変換されている間、データはノードの入力から出力に転送される。ノードの入力からのデータの消費は、その入力において使用可能なデータが本当にある場合にのみ起こることができる。相応して、経路をふさいでいる先の計算からのデータが無い場合のみ、データが出力において作成されることができる（つまり、後続のノードは先のデータ項目を消費していなければならない）。しかし、何らかの場合には、古いデータが経路をふさいでいても、出力においてデータを作成することが可能である。出力における古いデータはその場合

、新しいデータに置き換えられる。

【００５０】

一般的な発火規則に対する仕様は通常、以下の条件を含む。

１）ノードが入力データを消費するための、ノードの各入力に対する条件

２）ノードが出力においてデータを作成するための、ノードの各出力に対する条件

３）ノードの機能を実行するための条件

【００５１】

条件は通常、入力データの値、入力あるいは出力における有効データの存在、入力に適用された機能の結果、あるいは機能の状態、に依存するが、原則として、システムが使用可能な任意のデータに依存してもよい。

【００５２】

システムのノード１０１から１０５に対する一般的な発火規則を確立することにより、専用制御経路を必要とすることなく、多様な型のプログラムを制御することが可能になる。しかし、発火規則により、いくつかの特別な場合には、制御フローを実現することが可能である。他の特別な場合は、発火規則のないシステムであり、ここで全てのノード１０１から１０５は、ノード１０１から１０５の全ての入力においてデータが使用可能である場合にのみ動作する。

【００５３】

発火規則の機能の特定の例は、併合ノードを通して挙げることができる。このノードにより、制御フローを必要とせずに、データのフローを制御することが可能である。併合ノードは通常、２つのデータ入力を有し、そのうちの１つからデータが選択される。さらに併合ノードは、どのデータ入力からデータを取り出すかを選択するために使用される制御入力を有する。最後に併合ノードは、選択された入力データ値が配信される１つのデータ出力を有する。

【００５４】

例えば、ノードが２つの入力、ＴおよびＦを有すると仮定する。ノードを制御している条件は入力Ｃ上で受信され、結果は出力Ｒにおいて提供される。以下の発火規則は、１つの入力においてのみデータが使用可能であっても、ノードの出力においてデータを作成する。この場合、例えばＣ＝１であれば、入力Ｆにおいてデータが存在する必要はない。つまり、ノードの入力においてデータを消費するための条件は、以下のものである。

（Ｃ＝１　ＡＮＤ　Ｔ＝ｘ）　ＯＲ　（Ｃ＝０　ＡＮＤ　Ｆ＝ｘ）

ここで、ｘは有効値の存在を意味する。

【００５５】

さらに、ノードの出力においてデータを提供するための条件は、

（Ｃ＝１　ＡＮＤ　Ｔ＝ｘ）　ＯＲ　（Ｃ＝０　ＡＮＤ　Ｆ＝ｘ）

であり、ノードの関数は、

Ｒ　＝　ＩＦ　（Ｃ＝＝１）　Ｔ　ＥＬＳＥ　Ｆ

である。

【００５６】

データ・フローを制御するための他の型のノードは、スイッチである。スイッチ・ノードは通常、２つの出力ＴおよびＦ、１つのデータ入力Ｄ、および１つの制御入力Ｃを有する。ノードは、データがデータ入力および制御入力において使用可能な場合、その出力の１つにおいてデータを提供する。入力からのデータを消費するための条件は、

Ｃ＝ｘ　ＡＮＤ　Ｄ＝ｘ

であり、出力においてデータを提供するための条件は、

Ｔ：Ｃ＝１　ＡＮＤ　Ｄ＝ｘ

Ｆ：Ｃ＝０　ＡＮＤ　Ｄ＝ｘ

であり、ノードの関数は、

Ｔ　＝　ＩＦ　（Ｃ＝＝１）　Ｄ

Ｆ　＝　ＩＦ　（Ｃ＝＝０）　Ｄ

である。

【００５７】

図１ｂは、データ・フロー・マシンにおけるデータのフローを制御するための併合およびスイッチ・ノードの使用を示している。この場合、データ・フロー・マシンは、以下の関数に従ってｓの値を計算する。

$$s = \sum_{i=1}^{n} f(x_i)$$

【００５８】

上記の推論に従い、全ての種類の可能性のあるノードに対する発火規則を確立することが可能である。例えば、真ゲート（１つのデータ入力Ｄ、１つの制御入力Ｃ、１つの出力Ｒ、および関数Ｒ＝ＩＦ（Ｃ＝＝１）Ｄ）、非決定性優先権併合（２つのデータ入力Ｄ１およびＤ２、１つの出力Ｒ、および関数Ｒ＝ＩＦ（Ｄ１）Ｄ１　ＥＬＳＥ　ＩＦ（Ｄ２）Ｄ２）、加算（２つのデータ入力Ｄ１およびＤ２、１つの出力Ｒ、および関数Ｒ＝Ｄ１＋Ｄ２）、複製（１つのデータ入力Ｄ、１つの制御入力Ｃ、１つの出力Ｒ、および関数Ｒ＝Ｄ）、そして、ブールストリーム（入力なし、１つの出力Ｒ、および関数

Ｒ＝ＩＦ（ｓｔａｔｅ＝＝ｎ）ｓｅｔ　ｓｔａｔｅ＝０，ｒｅｔｕｒｎ　１
ＥＬＳＥ　ｉｎｃｒｅｍｅｎｔ　ｓｔａｔｅ，ｒｅｔｕｒｎ　０

である。

【００５９】

しかし、ノードの機能とは独立して、その入力においてデータ処置した後、ノード１０５は、その出力において処理しているデータの最終値を提供する。この場合、５つの入力におけるデータは単一の出力におけるデータを作成した。

【００６０】

データ・フロー・マシンのセマンティクスをよく調べると、これらのセマンティクスは、ディジタル回路が動作する方法、特にレジスタ・トランスファ・レベル（ＲＴＬ）において動作する方法に非常に似ていることが解る。データ・フロー・マシンにおいて、データはアーク上に存在し、データ上に何らかの動作を実行する機能ノードによって、１つのアークから他のアークに渡される。ディジタル回路においては、データはレジスタに存在し、データ上に何らかの機能を実行する組合せ論理によって、レジスタ間を渡される。データ・フロー・マシンのセマンティクスとディジタル回路の動作との間にこのような緊密な類似性があるので、データ・フロー・マシンを直接ディジタル回路に実装することが可能である。このことは、データ・フロー・マシンを通るデータの伝播は、データ・フロー・マシンの動作を実行するための状態機械のようなシミュレーション装置を必要とせずに、ディジタル回路内に実装することができることを意味する。代わりに、データ・フロー・マシンは、ノードを組合せ論理に置き換え、アークをレジスタあるいは独立して並行してアクセスされることができる他の高速記憶素子に置き換えることによって、直接実装することができる。

【００６１】

このことの利点は主に、実行速度である。このような実装は、プロセッサあるいは他の状態機械を通した実装よりも、より高いレベルの並列処理を使用することができるであろう。それはより容易にパイプラインすることができ、並列処理はより細かい細分性を有するレベルとなることができる。データ・フロー・マシン自体を実装するために状態機械の使用を避けても、データ・フロー・マシンのノードが状態機械を含むことはできることに留意されたい。

【００６２】

本発明の代替記述は、特殊なレジスターノードがデータ・フロー・グラフの機能ノード間に挿入され、エッジは単純なワイヤとして実現されることである。解りやすくするために、用語は、機能ノード、レジスタ・ノードおよびエッジを使用するのではなく、ノード

は組合せ論理として、エッジはレジスタとして、本発明を説明する。

【実施例１】

【００６３】

図２は、本発明の第１の単純な実施例を示している。より具体的には、本図は、図１のデータ・フロー・グラフのハードウェア実装を概略的に示している。図１の機能ノード１０１から１０５は、ノード２０１から２０５に置き換えられ、これらのノードは図１のデータ・フロー・グラフにおいて定義される数学あるいは論理機能を実行する。この機能は、組合せ論理によって実行することができるが、例えば状態機械あるいは何らかのパイプライン装置によっても実行することができる。

【００６４】

図２において、ワイヤおよびレジスタ２０６から２１５あるいはフリップフロップのような高速並列データ記憶ハードウェアが、図１の異なるノード間の接続に置き換わっている。ノード２０１から２０５の出力において提供されたデータは、他のノード２０１から２０５への即時あるいは後の転送のために、レジスタ２０６から２１５に格納される。本図から理解されるように、レジスタ２１３により、ノード２０１および２０２からのデータがノード２０４において処理されている間、ノード２０３からの出力値を格納することが可能となる。異なるノード２０１から２０５間で使用可能なレジスタ２０６から２１５がない場合、いくつかのノードの入力におけるデータは、同じ経路上の先行ノードにおける異なる組合せ遅延のために、不安定（値が変化する）となるかもしれない。

【００６５】

例えば、第１の組のデータが（レジスタ２０６から２１０を介して）ノード２０１から２０３の入力において提供されたと仮定する。ノードにおける処理の後、データは、ノード２０１から２０３の出力において使用可能となる。ノード２０１および２０２はそれからデータをノード２０４に提供し、一方ノード２０３はデータをノード２０５に提供する。ノード２０５はまたノード２０４からのデータも受信するので、データは、ノード２０５に転送される前にノード２０４において処理されなければならない。データがノード２０４を通って伝播される前に、新しいデータがノード２０１から２０３の入力において提供された場合、ノード２０３の出力は変わったかもしれない。従って、ノード２０５の入力におけるデータは、もはや正しくない。つまり、ノード２０４によって提供されるデータは、ノード２０５によって提供されるデータと比較して、より前の時点からのものである。

【実施例２】

【００６６】

上記の推論は単純化された推論である。実際においては、異なるノードに提供されるデータが正しいことを保証するために、高度クロッキング方式、通信プロトコル、追加ノード／レジスタ、あるいは追加論理回路が必要である。問題に対する最も簡単な解決策が図３に示され、ここでは、追加ノード３１６およびその関連するレジスタ３１７がデータ経路中に挿入されている。ノード３１６は、ＮＯＰ（ノー・オペレーション）を実行し、従ってその入力において提供されるデータを変更しない。ノード３１６を挿入することによって、グラフの全てのデータ経路において同じ長さが得られる。その結果、２０３と２０５との間のアークが、今や２つの構成要素を保持することができる。

【実施例３】

【００６７】

他の方法が図４ａに示され、ここで各ノード４０１は、全ての時点において正しいデータを提供するための追加信号線を備えている。第１の追加線は「有効」信号４０２を運び、この信号は、先行ノードがそれらの出力において安定したデータを有することを示す。同様にノード４０１は、ノード４０１の出力におけるデータが安定したものである場合、「有効」信号４０３をデータ経路における後続のノードに提供する。この手続きにより、各ノードは、その入力におけるデータの状態を決定することができる。

【００６８】

さらに、第２の追加線は「機能停止」信号４０４を運び、この信号は、現行ノード４０１がその入力においてどんな追加データも受信する準備ができていないことを先行ノードに示す。同様に、ノード４０１もまた、「機能停止」線４０５をデータ経路における後続ノードから受信する。機能停止線を使用することにより、特定の経路におけるデータのフローを一時的に停止することが可能である。このことは、ノードが、ある時間事例において、ループあるいはメモリ・アクセスのような時間のかかるデータ処理を実行し、不確定な遅延を生ずる場合に重要である。機能停止信号の使用は単に、本発明の１つの実施例である。選択されたプロトコルにより、いくつかの他の信号を使用することができる。例として、「消費済みデータ」、「受信可」、「応答」、あるいは「非応答」信号、および高あるいは低信号ではなくパルスあるいは遷移に基づく信号、がある。他の信号方式もまた可能である。「有効」信号の使用により、アーク上にデータが存在することあるいは存在しないことを表すことができる。従って、同期データ・フロー・マシンを構築することが可能なだけでなく、静的および動的データ・フロー・マシンも構築することができる。「有効」信号は必ずしも専用信号線として実現される必要はなく、「ヌル」値を表すための特別なデータ値を選択するような、いくつかの他の方法においても実装することができる。機能停止信号に関しては、多くの他の可能な信号方式がある。解りやすくするために、本文書の残りの部分では、機能停止および有効信号にのみ言及する。本発明の機能を他の信号方式に拡張することは簡単である。

【００６９】

特定の機能停止信号があれば、より効率を高めることができる。機能停止信号により、ノードは、下位のアークがその時点において一杯であっても、次のクロックサイクルにおいて出力トークンを受け取ることができることを知ることができる。機能停止信号がなければ、ノードは、発火する前に、下位のアーク上に有効データがなくなるまで待たなければならない。このことは、アークは、少なくとも１サイクルおきに空であることを意味し、従って効率が悪くなる。

【００７０】

図４ｂは、ノード４０１に対して、有効４０２、４０３および機能停止４０４、４０５信号を生成するための論理回路の１例を示している。本図に示される回路は、データが全ての入力上で使用可能な場合に発火するノードにおいて使用される。一般に、発火規則はより複雑であってもよく、個別のノード４０１の機能に従って確立されなければならない。

【００７１】

図４ｃは相応して、データ・フロー・マシンにおけるノード間のレジスタ４０６において使用される論理回路の１例を示している。この回路は、宛先ノードがまだデータを受け取る準備ができていない場合、レジスタがそのデータを保持し、このことをソースノードに信号を送って知らせることを保証する。回路はまた、レジスタが空の場合、あるいは宛先ノードがレジスタの現在の内容を受け取ろうとしている場合、新しいデータを受け取る。本図においては、解りやすくするために、１つのデータ入力４０７および１つのデータ出力４０８のみが示されている。しかし、入力および出力の実際の数は、システムのバス幅（つまり、トークンが何ビット幅であるか）によることを強調しておく。

【００７２】

複合データ・フロー・マシンの場合において、機能停止線は、信号伝播速度と比較して非常に長くなるかもしれない。このことは、機能停止信号が、機能停止される必要がある経路にある全てのノードに到達せず、結果としてのデータが失われる（つまり、まだ処理されていないデータが新しいデータによって上書きされる）という結果になるかもしれない。

【００７３】

この状況を解決するためには２つの一般的な方法がある。１つの方法は、機能停止信号伝播経路を非常に注意深く均衡がとれるようにし、機能停止信号が全ての目的レジスタに時間に間に合うように到達することを保証することである。他の方法は、先入れ先出しバ

ッファを停止可能なブロックの後に置き、ブロック内の機能停止信号の使用を完全に避けることである。代わりに、先入れ先出しが、データがパイプラインを出る時にパイプライン・データを収集するために、使用される。前者の解決策は、大規模なパイプラインされたブロックのために実装するには非常に難しく時間がかかる。後者は、ブロック内に存在する可能性のある全ての組のデータを保持することができる大きなバッファを必要とする。

【００７４】

この限られた信号伝播速度という問題を解決するより良い方法は、図６に示される「機能停止カッタ」と呼ばれる特徴による。機能停止カッタは基本的に、後続のノードから機能停止線を受信し、それを１サイクル遅らせるレジスタである。このことは、その点での機能停止信号の組合せ長をカットする。機能停止カッタが有効な機能停止信号を受信すると、それは先行ノードからのデータを１処理サイクルの間バッファに入れ、同時に、機能停止信号を同じ分遅らせる。機能停止信号を遅らせ入力データをバッファに入れることにより、非常に長い機能停止線が使用された場合であっても、データは失われないことが保証される。

【００７５】

機能停止カッタは、データ・ループ、特にパイプラインされたデータ・ループの実現を非常に簡単にすることができる。この場合、データのフローを制御するための多様なプロトコルは、機能停止信号が、ループを通るデータと同じ経路をしばしば逆にとることを要求する。このことは、機能停止信号に対する組合せループを生成する。機能停止カッタをループ内に置くことによって、このような組合せループを避けることができ、そうでなければ実現が困難あるいは不可能であろう多くのプロトコルが可能となる。

【００７６】

最後に、機能停止カッタは、データ・フロー・マシンにおけるデータ伝播の観点からは透過である。このことは、機能停止カッタを、自動的に必要な場所に追加することができることを意味する。

【実施例４】

【００７７】

図５ａは、本発明のさらに他の実施例を示している。ここで、グラフにおけるデータ経路は、ノード併合により等しくされている。グローバル・クロック信号を使用する設計に対しては、可能な限り高いクロック周波数は、最も遅い処理装置によって決定される。このことは、より高い周波数において動作する能力を有する全ての処理装置が、最も遅い装置によって設定される周波数において動作するよう制限されることを意味する。この理由により、どの装置も他の装置を遅くすることがないように、等しいあるいはほぼ等しい大きさの処理装置を獲得することが望ましい。グローバル・クロック信号を使用しない設計に対してであっても、分岐した計算における２つのデータ経路が等しい長さを持つ、つまり各データ経路に存在するノードの数が同じであることは有益である。データ経路が等しい長さであることを保証することによって、２つの分岐における計算が同じ速度で実行される。

【００７８】

図５ａに見られるように、図３の２つのノード３０４および３０５は、１つのノード５０４に併合されている。上記のように、このことは、異なるデータ経路の長さを等しくするために、あるいは設計の全体的な処理速度を最適化するために行ってもよい。

【００７９】

ノード併合は、いくつかのノード間のレジスタを除去することによって実行される。ここで、併合されたノードがより大きくなるにつれ、ノードの数は減少する。選択されたノードを組織的に併合することにより、ノードの組合せ深度は実質的に等しくなり、異なるノード間の処理速度は等しくされる。

【００８０】

ノードが併合されると、それらの個別の機能もまた併合される。このことは、異なる論

理素子を中間レジスタ無しに接続することによって行われる。ノードが併合されると、要求された場合にノードがそれらの出力においてデータを提供するために、新しい発火規則が決定されなければならない。

【００８１】

より具体的には、図５ｂに示されるように、２つのノード５０７、５０８を併合する場合、新しいノード５０９は、元のノードが有していた入力および出力アークの数から、組み合わされた２つのノード５０７、５０８を接続していたアークを引いたものと同じ数の入力および出力アークを有するよう作成される。上記のように、加算、乗算、等のような基本的な機能ノードに対して、発火規則は、全ての入力上にデータがある場合に発火し、全ての出力はデータを受信するために開放されている（以下にｎｍ発火規則と呼ばれる発火規則）。２つのこのようなノード５０７、５０８を併合した結果、３つの入力および１つの出力を有する新しいノード５０９ができる。加算からの２つの入力、乗算からの２つの入力、および２つのノード間の接続において使用される１つの入力は、併合されたノードに対する３つの入力となる。加算からの１つの出力、乗算からの１つの出力、および２つのノードを接続するのに使用される１つの出力は、併合されたノードからの１つの出力となる。併合されたノードに対する発火規則は、発火するためにその３つの入力全てにおけるデータを要求する。実際、ｎｍ発火規則を有するノードのどんな併合も、それ自体ｎｍ発火規則を有するが、入力および出力の数は変化しているであろう。元の２つのノード５０７、５０８の機能は、先にそれらを接続していたアークに従って、第１の組合せブロックからの出力を、他の組合せブロックの入力に直接接続することによって併合される。先にノード間のアークを表していたレジスタは除去される。その結果、より大きい組合せブロックとなる。

【００８２】

従って、例えば算術関数を実行するノードのような、その入力において常にデータを要求し、その出力において常にデータを提供するノードに対して、併合されたノードに対する発火規則は、元のノードに対するものと同じものとなる。

【００８３】

上記のように、機能プログラミング言語の使用は、データ・フロー・マシンにおいて超並列処理を達成するために不可欠である。本発明により、副作用に関する問題は、トークンにより処理される。インスタンス・トークンと呼ばれる特殊なトークンを使用することにより、副作用への可能性のあるアクセスの数、同様に、これらのアクセスが発生する順序を制御することができる。

【００８４】

副作用を使用しようと欲する全てのノードは、通常のデータ入力とは別に、問題となっている副作用に関連するインスタンス・トークンのための専用データ入力を持たなければならない。インスタンス・トークンのためのデータ入力とは別に、それはまた、インスタンス・トークンのための出力も持たなければならない。インスタンス・トークンのためのデータ経路は、データ・フロー・マシンにおける他のデータ経路として機能する。つまり、ノードは、その特定の動作を実行することができる前に、全ての適切な入力上にデータを持たなければならない。

【００８５】

副作用へのアクセスを必要とするノードに対する発火規則は、ノードがそのインスタンス・トークン入力上にデータ（つまり、インスタンス・トークン自体）を持たなければならないということである。副作用へのアクセスが完了すると、ノードはその出力においてインスタンス・トークンを解放する。この出力は次ぎに、同じ副作用へのアクセスを必要とする後続のノードのインスタンス・トークン入力に接続されてもよい。この手続きにより、インスタンス・トークン経路が、特定の副作用へのアクセスを必要とする全てのノード間で確立される。インスタンス・トークン経路は、ノードが副作用へのアクセスを得る順序を決定する。

【００８６】

従って、特定の副作用（例えば、メモリあるいは標識）に対して、そのインスタンス・トークン経路に沿って動いている１つあるいは複数のインスタンス・トークンがある。一連の全てのノードは、副作用へのアクセスを得るためにその入力上にデータを有する必要があるので、インスタンス・トークン・データ経路上のデータ要素の数を制限する（つまり、インスタンス・トークンの数を制限する）ことにより、副作用への同時アクセスの数を制限することが可能である。１つのインスタンス・トークンのみが、特定の時点においてインスタンス・トークン経路上に存在するよう許される場合、副作用は、同時に２つあるいはそれ以上のノードからアクセスされることはないことが保証される。さらに、副作用がアクセスされる順序は、インスタンス・トークン経路によって明確に決定される。一定の状況の間１つ以上のノードが副作用へのアクセスを得るようすることが安全な場合、１つ以上のインスタンス・トークンを同時に経路に導入することが可能である。インスタンス・トークン経路を分割し、インスタンス・トークンを分割の双方の経路に複製することも安全かもしれない。

【００８７】

例えば、副作用としてメモリにアクセスする時、双方の経路がメモリからの読出しのみを含む場合、通常インスタンス・トークン経路を分割することが安全であろう。この場合、メモリへの同時アクセスは、メモリ制御装置によって任意に決定されるが、読出しに対する実行の順序は互いに影響しないので、このことは安全である。対照的に、２つの経路が書込みを含む場合、２つの書込みが実際に実行された順序は必須である。それは、メモリが最終的にどの値を保持するかを決定するからである。このような場合、インスタンス・トークン経路は、安全に分割することができない。

【００８８】

一本のインスタンス・トークン経路上にいくつかのインスタンス・トークンを続けて置くことは、通常、パイプライン化された計算の異なる「世代」によるメモリへのアクセスを表すであろう。例えば、２つの世代が、それらがメモリの同じ部分をアクセスしないことにおいて、関連性がないと知られている場合、多数のインスタンス・トークンを続けて挿入することは安全である。

【００８９】

いくつかの異なる副作用（例えば、メモリあるいは他の入力あるいは出力装置）へのアクセスを続けて置くこともまた可能である。このことには、経路上の各インスタンス・トークンに対して、各副作用へのアクセスの順序を明確に決定するという効果がある。例えば、入力装置からの読出しは、インスタンス・トークン経路上において、出力装置への書込みの前に置くことができる。いくつかのインスタンス・トークンが経路上に同時に存在する場合、読出しおよび書込みに対する全体的な順序は決定的でないかも知れないが、経路上の各個別のインスタンス・トークンに対しては、副作用間で明確な順序付けがあるであろう。

【００９０】

最後に、ディジタル回路を設計する場合、異なる型のデータ・フロー・マシンを混合させることができる。例えば、データ依存回数の反復を伴うループは、その部分以外は静的である静的データ・フロー・マシン内の動的データ・フロー・マシン部分とすることができる。このように、反復は並行して実行することができ、このことは静的データ・フロー・マシンにはできないことである。静的データ・フロー・マシンのこのような局所的動的部分は、動的データ・フロー・マシンの完全なタグ・マッチング・システムが無くても動作することができる。代わりに、トークンが、それらが入ったのと同じ順序で動的部分を出て行くことを保証しさえすればよい。マシンの残りの部分は静的でありトークンを再順序付けしないので、このことはトークンをマッチさせる。

【００９１】

再帰に入る各トークンに通し番号をタグ付けし、順序を外れて再帰を終了するトークン収集するためにバッファを使用することによって、再帰が終了した後にトークンを正しい順序に再配置することが可能である。より具体的には、バッファは再帰ステップの後に配

置される。トークンが順序を外れて再帰を抜ける場合、トークンは、より小さい通し番号を持つ全てのトークンが再帰を抜けるまでバッファ内に置かれる。従って、バッファの大きさが、いくつのトークンが順序を外れて再帰を抜けることができるかを決定し、トークンを再帰の完了の後に正しく配置することができることを保証する。一定の場合、再帰を抜けるトークンの順序が適切でないことを知ることができる。例えば、再帰を抜けるトークンの値の単純な合計が実行される場合である。このような場合、データ・トークンへの通し番号のタグ付けおよびバッファの双方が省略されてもよい。

【００９２】

データ依存ループは別として、ローカル・タグ・マッチングおよび再順序付け方式の使用は、他の型の再順序付けノードあるいは副グラフに対しても使用することができる。

【００９３】

本発明は、好ましい実施例を参照しながら上に説明されてきた。しかし、ここに開示される以外の実施例も、付随する請求項によって定義される本発明の範囲内において可能である。

【図面の簡単な説明】

【００９４】

【図１ａ】それ自体知られている第１のデータ・フロー・グラフを示す概略図である。

【図１ｂ】それ自体知られている第２のデータ・フロー・グラフを示す概略図である。

【図２】本発明の第１の実施例を示す図である。

【図３】異なるデータ経路の長さが等しくされた、本発明の第２の実施例を示す図である。

【図４ａ】本発明の第３の実施例によるノードの詳細な概略図である。

【図４ｂ】発火規則を確立するための論理回路の１例を示す図である。

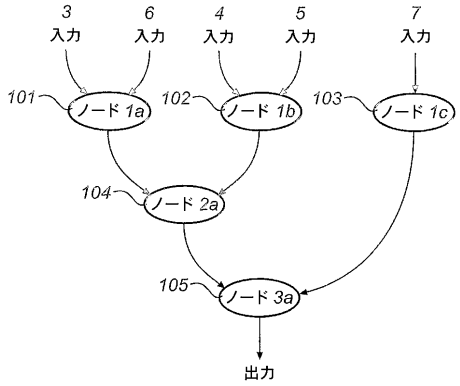
【図４ｃ】相応して、データ・フロー・マシンにおけるノード間のレジスタにおいて使用される論理回路の１例を示す図である。

【図５ａ】異なるデータ経路の長さがノード併合によって等しくされた、本発明の第３の実施例を示す図である。

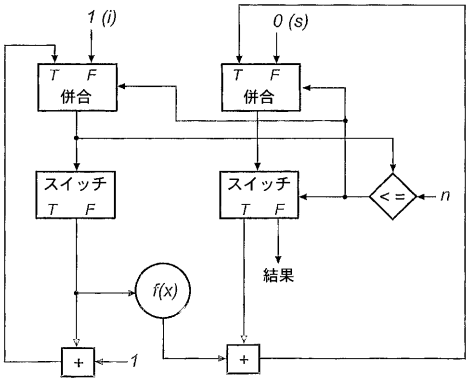
【図５ｂ】図５ａにおける２つのノードの併合のより詳細を示す図である。

【図６】本発明による機能停止カッタの１つの実施例を示す図である。

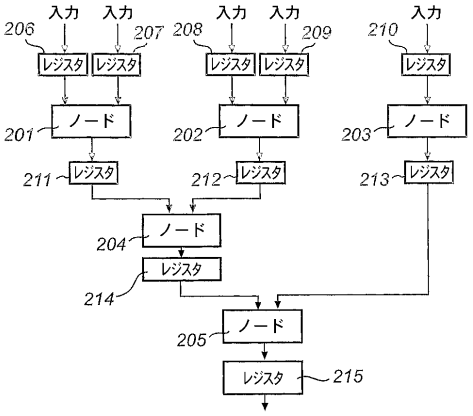
【図１ａ】



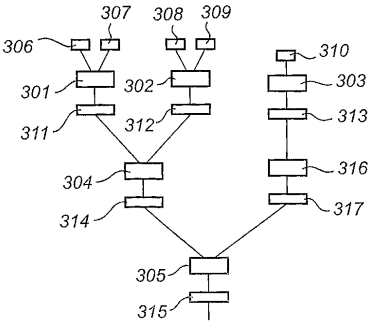
【図１ｂ】



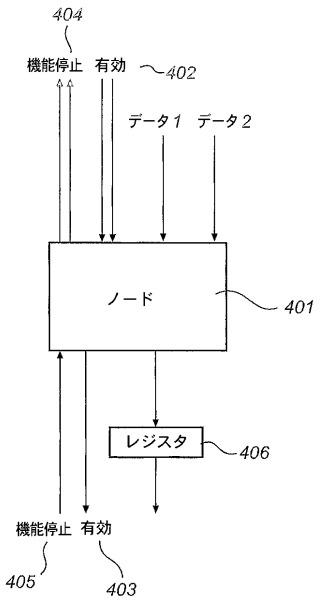
【図２】



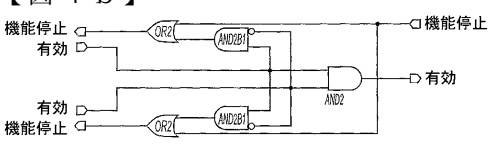
【図３】



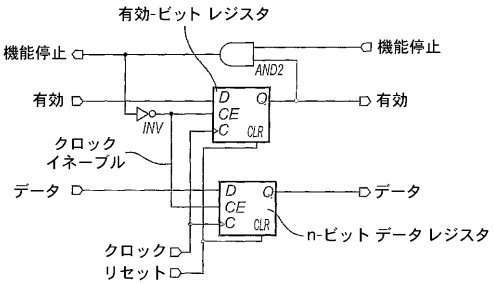
【図４ａ】



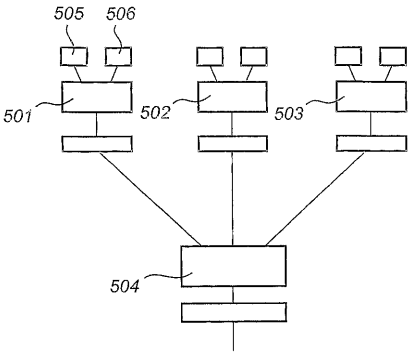
【図４ｂ】



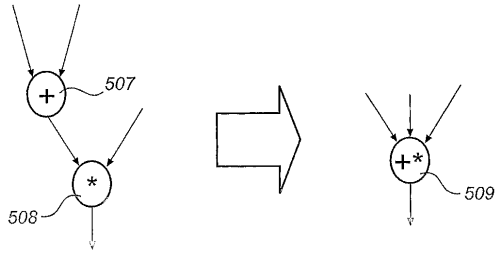
【図４ｃ】



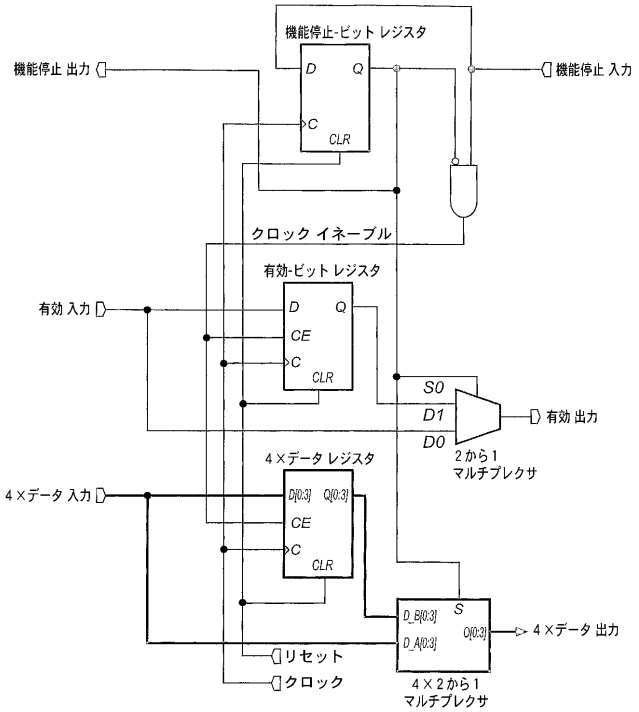
【図５ａ】



【図５ｂ】


507
508
509

【図６】


機能停止-ビット レジスタ
機能停止 出力
機能停止 入力
D
Q
C
CLR
クロック イネーブル
有効-ビット レジスタ
有効 入力
CE
S0
D1
D0
有効 出力
2から1
マルチプレクサ
4×データ レジスタ
4×データ 入力
D[0:3]
Q[0:3]
S
D_B[0:3]
D_A[0:3]
4×データ 出力
リセット
クロック
4×2から1
マルチプレクサ

【国際調査報告】

INTERNATIONAL SEARCH REPORT

International application No.
PCT/SE 2004/000394

A. CLASSIFICATION OF SUBJECT MATTER

IPC7: G06F 15/82, G06F 9/44
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)

IPC7: G06F

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

SE,DK,FI,NO classes as above

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

EPO-INTERNAL, WPI DATA, PAJ

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	VERDOSCIA, L et al: Conditional and itertive structures using a homogenous static dataflow graph model. In: Massively Parallel Computing Systems, 1994, Proceedings of the First International Conference on. May 1994. Pages 391-401. Inspec Accession Number: 5050124	1-29
A		38-47,77
	--	
X	US 5021947 A (CAMPBELL, M J), 4 June 1991 (04.06.1991)	1-29
A		38-47,77
	--	

[X] Further documents are listed in the continuation of Box C. [X] 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
8 June 2004	01-12-2004
Name and mailing address of the ISA/ Swedish Patent Office Box 5055, S-102 42 STOCKHOLM Facsimile No. +46 8 666 02 86	Authorized officer Oskar Pihlgren /LR Telephone No. +46 8 782 25 00

Form PCT/ISA/210 (second sheet) (January 2004)

CORRECTED

INTERNATIONAL SEARCH REPORT

International application No.
PCT/SE 2004/000394

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	JUNG,B ET AL: Node merging: A transformation on bit-level dependence graphs for efficient VLSI array design. In: Application-Specific Array Processors, 1993. Proceedings., Internatinal Conference on, 25-27 October 1993, Venice, Italy. On pages 442-453. --	30-37,75,76
A	CHATTERJEE,M ET AL: Buffer assignment algorithms on data driven ASICs. In: Computers, IEEE Transactions on, Jan. 2000. On pages 16-32, vol.49, Iss.1, ISSN: 0018-9340. --	48-51
A	US 4943916 A (ASANO, H ET AL), 24 July 1990 (24.07.1990) --	52-74
A	US 4841436 A (ASANO, H ET AL), 20 July 1989 (20.07.1989) -- --------	52-74

Form PCT/ISA/210 (continuation of second sheet) (January 2004)

CORRECTED

INTERNATIONAL SEARCH REPORT

International application No.
PCT/SE 2004/000394

Box No. II Observations where certain claims were found unsearchable (Continuation of item 2 of first sheet)

This international search report has not been established in respect of certain claims under Article 17(2)(a) for the following reasons:

1. ☐ Claims Nos.:
because they relate to subject matter not required to be searched by this Authority, namely:

2. ☐ Claims Nos.:
because they relate to parts of the international application that do not comply with the prescribed requirements to such an extent that no meaningful international search can be carried out, specifically:

3. ☐ Claims Nos.:
because they are dependent claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a).

Box No. III Observations where unity of invention is lacking (Continuation of item 3 of first sheet)

This International Searching Authority found multiple inventions in this international application, as follows:

see extra sheet

1. ☐ As all required additional search fees were timely paid by the applicant, this international search report covers all searchable claims.
2. ☒ As all searchable claims could be searched without effort justifying an additional fee, this Authority did not invite payment of any additional fee.
3. ☐ As only some of the required additional search fees were timely paid by the applicant, this international search report covers only those claims for which fees were paid, specifically claims Nos.:

4. ☐ No required additional search fees were timely paid by the applicant. Consequently, this international search report is restricted to the invention first mentioned in the claims; it is covered by claims Nos.:

Remark on Protest

☐ The additional search fees were accompanied by the applicant's protest.

☐ No protest accompanied the payment of additional search fees.

Form PCT/ISA/210 (continuation of first sheet (2)) (January 2004)

CORRECTED

INTERNATIONAL SEARCH REPORT

International application No.
PCT/SE 2004/000394

Continuation of Box III:

The application contains five different inventions a priori, with no special technical features in common:

1. Claims 1-29 describe a method, an apparatus and a data flow machine for implementing digital logic circuitry for a graph representation comprising functional nodes, where hardware elements are specified for each functional node and each connection between them, and a firing rule is specified for each functional node.

2. Claims 30-37, 75 and 76 describes a method and an apparatus for implementing digital logic circuitry for a graph representation comprising functional nodes, where a merged hardware element is specified.

3. Claims 38-47 and 77 describes a method and a data flow machine where hardware elements are adapted to be enabled and disabled for activation.

4. Claims 48-51 is described a method and an apparatus for ensuring data integrity in a data flow machine.

5. Claims 52-74 describes a method, an apparatus and a data flow machine for implementing digital logic circuitry for a graph representation comprising functional nodes, where a re-ordering hardware element is specified.

CORRECTED

INTERNATIONAL SEARCH REPORT

Information on patent family members

30/10/2004

International application No.
PCT/SE 2004/000394

US	5021947	A	04/06/1991	EP	0261173	A 30/03/1988
				IL	81756	D 00/00/0000
				JP	63503099	T 10/11/1988
				WO	8706034	A 08/10/1987
US	4943916	A	24/07/1990	JP	1835688	C 11/04/1994
				JP	2016236	Y 02/05/1990
				JP	5046593	B 14/07/1993
				JP	61276032	A 06/12/1986
				JP	62028856	U 21/02/1987
US	4841436	A	20/07/1989	JP	1901372	C 27/01/1995
				JP	6032056	B 27/04/1994
				JP	61276031	A 06/12/1986
				JP	62029275	U 21/02/1987

Form PCT/ISA/210 (patent family annex) (January 2004)

CORRECTED

フロントページの続き

(81)指定国 AP(BW,GH,GM,KE,LS,MW,MZ,SD,SL,SZ,TZ,UG,ZM,ZW),EA(AM,AZ,BY,KG,KZ,MD,RU,TJ,TM),EP(AT,BE,BG,CH,CY,CZ,DE,DK,EE,ES,FI,FR,GB,GR,HU,IE,IT,LU,MC,NL,PL,PT,RO,SE,SI,SK,TR),OA(BF,BJ,CF,CG,CI,CM,GA,GN,GQ,GW,ML,MR,NE,SN,TD,TG),AE,AG,AL,AM,AT,AU,AZ,BA,BB,BG,BR,BW,BY,BZ,CA,CH,CN,CO,CR,CU,CZ,DE,DK,DM,DZ,EC,EE,EG,ES,FI,GB,GD,GE,GH,GM,HR,HU,ID,IL,IN,IS,JP,KE,KG,KP,KR,KZ,LC,LK,LR,LS,LT,LU,LV,MA,MD,MG,MK,MN,MW,MX,MZ,NA,NI,NO,NZ,OM,PG,PH,PL,PT,RO,RU,SC,SD,SE,SG,SK,SL,SY,TJ,TM,TN,TR,TT,TZ,UA,UG,US,UZ,VC,VN,YU,ZA,ZM,ZW

(72)発明者 メール、ステファン
スウェーデン国、ルンド、マギストラトスベーゲン ２１ エイ
(72)発明者 ボルグ、ポントゥス
スウェーデン国、ルンド、オレ レメルス ベーグ １６、 イデオン、フロー コンピューティング エービー気付