依存性グラフを用いた非同期式回路合成システム

9M－6

籠谷裕人　Hong Minh Nhut　南谷 崇
東京工業大学 工学部

1 はじめに

近年の素子技術はスイッチング遅延が1ピコ秒にせまる高速なデバイスを実現しつつある。しかし従来の同期式プロセッサ回路はチップ全体へのクロック分配が必要であり、配線遅延の相対的な増大によるクロックスキューのため、こうした素子を活用できるような高速のクロックを用いることができない[1]。

素子の高速性を有効に活用する一つの方法は、プロセッサを非同期式回路で構成することである。非同期式回路は、同期式回路の設計にあるような論理設計とチップ設計の相互依存性を排除でき、また、回路を拡張する場合のタイミング設計のやり直しも不要となり拡張性に富むといった利点を持つ。

我々は文献[2]で、非同期式制御回路の自動合成手法を提案した。本稿ではまずその概要について述べた後、データパスにおけるレジスタや演算回路の接続を合成する手順を示す。またこの手法を用いた合成システムを試作したので、その構成の概要について述べる。

2 非同期回路モデル

本合成方式で対象とする非同期式システムは、図1に示すように、制御回路とデータパスに分離される。クロック信号は存在しないため、制御回路とデータパスの間は、要求信号と応答信号からなるハンドシェークによって接続される。

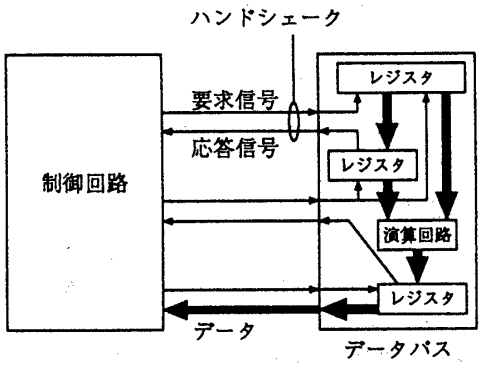


図 1: 回路のモデル

一つのハンドシェークは、一つの基本的な処理の実行を制御する。すなわち、要求信号が1になるとデータパス内の必要なレジスタからデータが出力され、そのままのデータや演算を施された結果を別のレジスタに転送し、転送先のレジスタから応答信号の1を返す。さらにその応答信号を見て要求信号が0に戻り、各データの出力が止まって応答信号を0にする。データ信号は、データの有無を判断できるように二線符号などに符号化される。

以上のような、一つのハンドシェークで制御される基本的な処理の単位を、基本操作と呼ぶ。

Synthesis System of Asynchronous Circuits using Dependency Graph
Hiroto Kagotani, Hong Minh Nhut, Takashi Nanya
Faculty of Engineering, Tokyo Institute of Technology

3 仕様記述と制御回路の合成

3.1 仕様記述方式

システムの動作仕様は、基本操作を基本的な単位として記述される。つまり一般的なディジタル回路設計におけるレジスタ・トランスファ・レベルの記述となる。

一方、システムが実現すべきアルゴリズムや、ハードウェア上の制限によって、そのシステムでの基本操作の実行順序には制約が存在する。これを依存性と呼ぶ。基本操作の間の依存性を記述することによって、その順序や並列性を表すことができる。

また一般にアルゴリズムの記述には条件判断やループなどが必要となる。そのため依存性グラフの構成要素としてフォークおよびジョインというノードを用いる。以上をまとめたものが図2(a)となる。これらのノードを矛盾なく接続してグラフとして記述したものが、本合成方式で用いられる仕様となる。

図4(a)は、プロセッサの命令フェッチ動作を依存性グラフで記述した例である。

3.2 制御回路の合成

依存性グラフによる仕様に基づいてハンドシェークを行うには、その依存性グラフの処理の流れを回路で模擬すればよい。すなわちグラフに現れる各種の構成要素を、その意味を変化させないように回路ブロックに置き換えると制御回路が合成できる。

具体的には、基本操作を表すノードは、その実行に必要なハンドシェークの制御を行うQモジュール(図3)に対応する。また制御の待ち合わせを表すノードは、C素子に対応する。これら以外についても図2(b)のように対応させる。

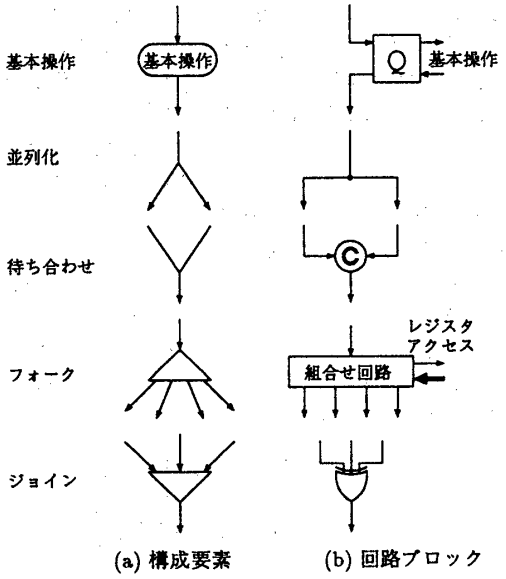


図 2: 依存性グラフからの制御回路合成

このように依存性グラフに対応して制御回路を合成すると、前の処理が完了したことを入力信号の1によって知らされるため、クロック信号を用いずに基本操作の実行タイミングを制御できる。

以上の手順により、図4(a)の命令フェッチ動作の例では、図4(b)のように制御回路が合成される。

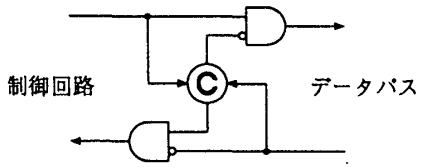
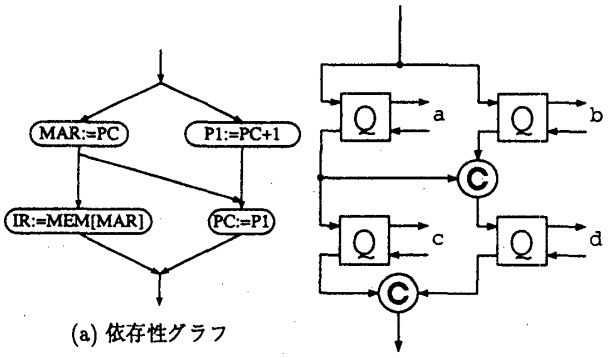


図 3: Q モジュール



(a) 依存性グラフ

(b) 合成された制御回路

図 4: 命令フェッチ回路による例

4 データパスの合成

4.1 合成手順

与えられた依存性グラフからデータパスを合成するには、そのグラフに含まれる基本操作の集合がもとになる。

ある基本操作について、参照や更新に用いられるレジスタと、演算回路を用意する。参照の場合はそのレジスタにデータ出力ポートを付加し、ハンドシェークの要求信号を出力要求信号に接続する。更新の場合はデータ入力ポートを付加し、更新されるレジスタの入力応答信号を一つの C 素子でまとめてハンドシェークの応答信号とする。更新されるレジスタが一つの場合は C 素子は用いない。さらに、各データ出力線とデータ入力線を演算回路に接続する。データの転送のみの場合は演算回路は不要である。

以上を先の命令フェッチ動作について $P1 := PC+1$ の例で示す。まずレジスタ $PC, P1$ と演算回路 [+1] を用意する。PCにはデータ出力ポート、$P1$ にはデータ入力ポートを付加する。PCの出力要求信号は制御回路からの要求信号が接続され、$P1$ の入力応答信号は制御回路への応答信号となる。また PCからのデータ出力は [+1] を介して $P1$ のデータ入力に接続される。

このような処理を各基本操作について繰り返すと図5のような回路が得られる。この回路のレジスタやメモリについては既にゲートレベルで定義されている。

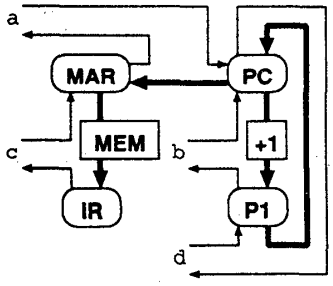


図 5: 命令フェッチ回路のデータパス

4.2 演算回路の共有

以上のデータパス合成だけでは、回路の共有などを行わないため非常に冗長な回路が得られる。そこで、まず依存性グラフ中に同一の基本操作が複数存在する場合、[2] で示す回路モジュールによってハンドシェークのマージを行う。さらに共有可能な同一の演算は、マルチプレクサによって共有する。

5 合成システムの構成

上で示した合成手法を実現するシステムを計算機上に試作した。

本システムではグラフィカルなユーザインタフェースは用意されていない。そのため、依存性グラフはノードとその接続関係をテキストで記述して入力する。またデータの演算を行う組み合せ回路については、別に与えられるものと仮定し、ブラックボックスとして扱っている。

出力形式は、ハードウェア記述言語として一般的である Verilog 形式のネットリストである。これは、将来シミュレーションによる評価などを容易にするためである。

図 6は、回路の合成システムの構成を示すが、現在のシステムはこのうち実線の部分のみを実現している。本システムは C 言語で作成され、さまざまな機種で動作する。

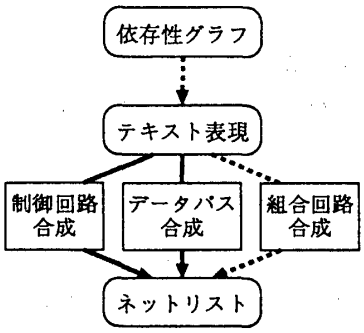


図 6: 自動合成システムの構成

6 まとめ

依存性グラフを仕様として非同期式回路を自動合成する方式を示した。またこの方式で合成を行うシステムを計算機上に試作し、その概要について述べた。

複数の依存性グラフからなる仕様をもとに回路システムを合成する方式の検討や、その方式の実装は今後の課題である。また割り込み処理や演算回路の合成、回路の最適化についても検討したい。

本研究の一部は文部省科学研究費補助金 04452192 による。

参考文献

[1] Takashi Nanya. Challenges to dependable asynchronous processor design. In *Proc. Int. Symp. on Logic Synthesis and Microprocessor Architecture*, pp. 132–139, July 1992.

[2] 綿谷裕人, 南谷崇. 依存性グラフに基づく非同期式制御回路の合成. 信学技報, October 1992. FTS92-16.