計算機アーキテクチャ 110－5
設 計 自 動 化 73－5
（1995．1．19）

分割操作による論理回路の トップダウン設計手法について

大和 明宏† 新井 浩志‡ 深澤 良彰†

†早稲田大学理工学部 ‡千葉工業大学工学部

概要

機能ブロックレベルから、論理回路を設計するための手法として、分割操作によるトップダウン設計手法を提案する。

機能ブロック図を用いたトップダウン設計は、従来の回路図エディタなどを用いておこなうことも可能である。しかし、ブロック間の機能的関係が明確に規定されていないため、ブロック間の関係が不明瞭になる可能性がある。2つのブロック間の機能的関係は、その間の信号の流れ方によって規定されると考えられる。本設計手法においては、ブロック間の信号の流れに従って、ブロック詳細化操作を3種類に限定する。これによって、理解し易い論理回路の設計をおこなう。

本稿では、設計対象、モデル、および、設計操作を定義した上で、本稿で述べる接続制約を満たすような任意の論理回路を3種類の分割操作で設計可能であることを検証する。また、実際に本手法を用いて16ビット計算機などを設計した結果について報告する。

A Top Down Design Method with Dividing Operations

Akihiro Yamato † Hiroshi Arai ‡ Yoshiaki Fukazawa †

†School of Science and Engineering, Waseda University ‡Faculty of Engineering, Chiba Institute of Technology

Abstract

For designing gate-level logic circuits from functional-block-level diagrams, a top down method based on dividing operations is proposed.

By traditional schematic editors, it is possible to design logical circuits in top-down manner, however, the relations between functional blocks often become unclear because of the lack of any standards for block refinements. Functional relation between two blocks is considered to be determined by its signal flow. In our method, the block refinement operation is classified by how the signals flow between the two blocks. Thus, the designed circuits will become easy to understand.

In this paper, it is shown that our design method can be applied to any logic circuits with certain connective constraints. As examples, the results of designing a 16bits computer and so on with our method are also reported.

1 はじめに

大規模な対象物を設計するための手法として、トップダウン設計法が広く用いられている。トップダウン設計法では、複雑な設計問題を、より単純な部分問題の集合に分割しながら設計を進める。設計者は、個々の部分問題を、それ以外の部分との関係だけに注意しながら解決すれば良いため、大規模な問題を扱い易くなる。

大規模な論理回路の設計問題においても、設計の各段階におけるブロック間の機能的関係を明確にしながら、トップダウンに設計を進めることは有効である。近年は、ハードウェア記述言語と論理合成技術を用いた、トップダウン設計手法が普及し始めている[1]。しかし、機能ブロック図などを用いた表現方法なども広く使用されており、言語による表現と使い分けられているのが現状である。

我々は、トップダウン設計法を積極的に支援することを目的とし、回路図エディタ SCHET(助っ人： SCHematic Editor for Top-down design)[2],[3] に関する研究を行なっている。

本稿では、機能ブロック図レベルから、トップダウンに論理回路を設計するための手法として、分割操作によるトップダウン設計手法を提案する。

一般に広く用いられている回路図エディタ[4]においても、階層設計された図面を管理するための様々な機能が実用化されており，これらの機能を利用して論理回路をトップダウンに設計することができる．しかし，ブロック間の機能的関係を明確に規定していないため，抽象的なブロックを任意のブロック群に分割することが可能であり，ブロック間の機能的関係が不明確になる可能性がある．本トップダウン設計手法では，ブロック間の機能的な関係を，3種類の基本的な関係の組合せとして表現する．ハードウェア設計者は，これらの基本的な関係に対応したブロック分割操作を繰り返すことによって，抽象的なブロックを具体的なブロックに分割する．このように，基本的な詳細化操作を用いて段階的に詳細化することによって，ブロック間の機能的関係が不明確になることを防ぎ，大規模な論理回路の設計品質を向上させる．

本稿では、3種類の機能的関係だけを用いて設計することが可能な論理回路のクラスについて検討を加えた結果を報告する。以下、設計対象、モデル、および、設計操作を定義し、本稿で述べる接続制約を満たすような任意の論理回路を3種類の分割操作で設計可能であることを検証する。

2 分割操作によるトップダウン設計

トップダウン設計は、設計対象を徐々に詳細化していくことにより、各詳細化レベルでの部分間の整合性を管理し易いという利点を持つ。反面、ある程度設計した経験を持つ対象でないと適用が困難であるため、現実には、トップダウン設計とボトムアップ設計を組み合わせて用いていることが多い。以下本稿では、任意の論理回路をすべてトップダウンに設計すると仮定して述べる。しかし、ボトムアップ手法を併用して設計された機能ブロックをライブラリに保持し、そのブロックレベルまでの設計をおこなう場合にも適用可能である。

本稿における分割操作では、抽象的な機能ブロックを詳細化するための操作を考える。抽象的なブロックは、抽象的な入出力を持つものとする。この時、詳細化操作とは、入出力に関する詳細化と、ブロック内の機能の詳細化であると考えられる。

入出力に関する詳細化操作は、抽象的な入力信号、出力信号を具体化する操作であり、抽象的な線を分割することにより、具体的な信号の集合を定義することである。

ブロック内の機能の詳細化は、1つのブロックを複数のブロックによって実現する操作である。2つのブロック間の関係は、その間の信号の流れ方によって規定されると考えられる。従って、本分割手法においては、詳細化操作を、ブロック間の信号の流れに従って分類する。2つのブロック間での信号の流れは、逐次的に処理される場合と並列的に処理される場合に分けられる。さらに、フィードバック信号のように、単一のブロックでの自分自信への信号の流れが考えられる。よって、本研究では、逐次処理的ブロックへの分割 (レベル分割：図1(a))、並列処理的ブロックへの分割 (同等分割：図1(b))、そして、フィードバックをおこなうブロックへの分割 (ループ分割：図1(c)) の3種類の分割操作だけで論理回路を設計することを考える。

ブロック間のデータの流れに関してはこれで十分であると考えられるが、大規模な回路を考えた場合は、より多くの機能的関係を定義した方が設計が容易で、理解し易くなる可能性がある。その様な分割操作として、制御関係を表す分割操作などが考えられる [2]。しかし今回は、論理回路の設計可能性という視点から、三種類の分割操作だけで十分である事を検証する。

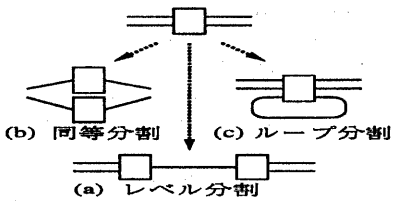


図 1: 3種類の分割操作のイメージ

図2に、上記の操作を用いて EDC[5] の実行制御用のパネルスイッチコントローラを設計した過程の一部を示す。まず、スイッチコントローラは、大きく実行スイッチ (RUN) とクリアスイッチ (CLR) という、2つから成っているので、同等分割で RUNCTL と CLRCTL の2つに分ける。次に、ブロック CLRCTL は、現状態から次状態が決まるので、ループ分割を用いてフィードバック信号を生成する。そして最後に、各ブロックをラッチとその値を用いる組合せ回路にレベル分割している。

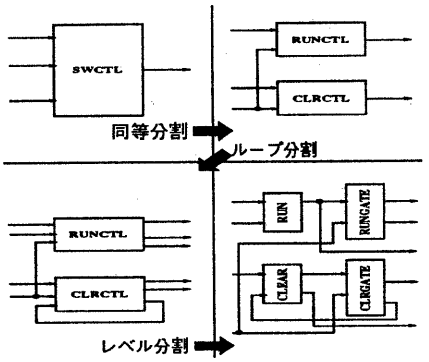


図 2: 設計途中経過の例

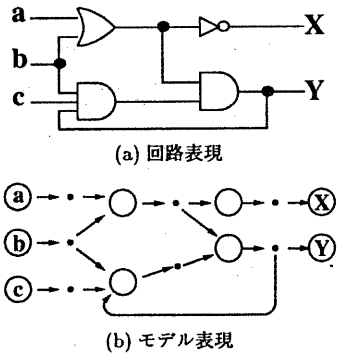


図 3: 回路例

3 準備

3.1 設計対象

本手法は、バスシステムに接続されるような個々の機能ブロックを設計対象と考える (現在の本手法の性質から、バスの設計は考えない)。すなわち、「信号の流れる方向が一定」かつ「論理素子の出力どうしが直接接続しない」という接続上の性質を満足する論理回路を設計対象とする。よって、以下に示すような素子は含まないものとする。

- トライステートバッファ (バス論理)
- ワイヤード OR やワイヤード AND 等 (結線論理)

論理回路を、ラベル付きの節点集合 V とラベル付きのアーク集合 A から成る有向グラフ $G = (V, A)$ によって表現する。V はブロックや入出力端子を表すノード集合 V_{node} と、信号線分岐可能点を表すドット集合 V_{dot} で構成される。さらに、V_{node} は入力端子集合 V_{in} 、出力端子集合 V_{out} と機能ブロック集合 V_{block} から成る。A はノードからドット、またはドットからノードへ向かう有限個のアーク (有向辺) の集合である。すなわち、$G = (V, A)$ は、ラベル付き 2 部有向グラフ [6] である。

3.2 接続制約

ブロックには必ず入出力線が存在する。また、本モデルにおいては、論理素子の出力どうしが直接接続せず、双方向の信号線、入出力端子は存在しないことから、節点に入力するアークの本数、出力するアークの本数をそれぞれ Arc_in、 Arc_out とした時に、次式 (1) を満足しなければならない。

$$\left.\begin{array}{ll} \mathrm{Arc\_in}(v) \geq 1, & \mathrm{Arc\_out}(v) \geq 1 \quad (v \in V_{block}) \\ \mathrm{Arc\_in}(v) = 1, & \mathrm{Arc\_out}(v) \geq 1 \quad (v \in V_{dot}) \\ \mathrm{Arc\_in}(v) = 0, & \mathrm{Arc\_out}(v) = 1 \quad (v \in V_{in}) \\ \mathrm{Arc\_in}(v) = 1, & \mathrm{Arc\_out}(v) = 0 \quad (v \in V_{out}) \end{array}\right\} \quad (1)$$

図 3.2に、回路例とそのモデル表現を示す。

4 分割操作

4.1 ブロック分割操作

3 章で説明した回路モデルに対する 3 種類の分割操作の定義を以下に記す。

(a) レベル分割 (level_div)

ノード v に対するレベル分割を $level\_div(v)$ で表す。この分割操作は、対象ノードを 2 つに分割し、分割されたノードの間に新たにアークとドットを付加する (図 4)。分割前のノードへの入出力アークを、分割後のどのノードへ接続するかは、設計者が指定する。

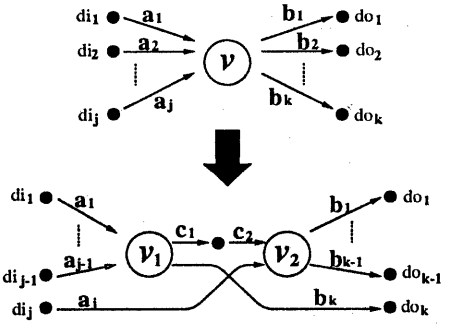


図 4: レベル分割

(b) 同等分割 (para_div)

ノード v に対する同等分割を $para\_div(v)$ で表す。同等分割操作は対象ノードを二つのノードに分割し、分割前のノードへの入出力アークを、分割後のどのノードへ接続するかを指定する。一つのアークが分割後の両ノードに接続する時には、そのアークが出るドットから新たにアークを付加する (図 5)。

(c) ループ分割 (loop_add, loop_div)

ノード v からドット d へ向かう、既存のアークをフィードバックさせる処理を loop_add(d) と定義する。また、新たなフィードバック信号線を付加する処理を loop_div(d_{new})

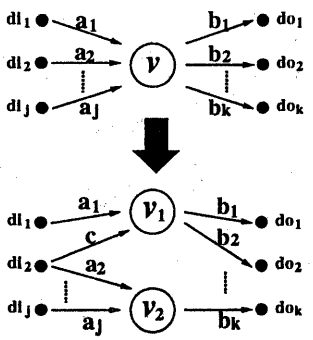


図 5: 同等分割

と定義する。loop_add では、対象ノードの出力アークの中から、フィードバックさせたいものを選び、そのアークのドットから対象ノードに対して新たにアークを付加する。loop_div ではアークとドットを付加し、新たなフィードバック信号線を追加する。(図 6)

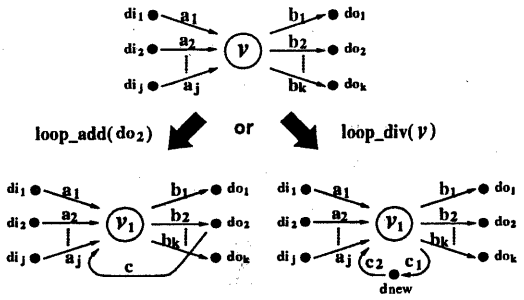


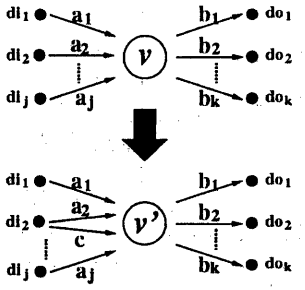
図 6: ループ分割

4.2 信号線分割操作

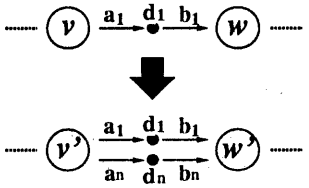
ブロック分割操作の他に、信号線を分割詳細化するための処理として、同一信号を分岐させる操作と、新たな信号線を生成するための操作の、2つを考える。

前者は、ドットからノードへ至るアークを増やすことを意味する (図 7(a))。また、後者はノードから他のノードへ至るアークとドットを増やすことを意味する (図 7(b))。なお、信号線分割において、ノードのラベル名が変化することはないが、図中では理解容易化のためにノードラベルに「'」をつけてある。

アーク a に対する同一信号分岐処理を arc_div(a) と定義する。また、ドット d と、d に結合しているアークに対して、新たな信号線を生成する操作を line_div(d) と定義する。



(a) arc_div(a_2)



(b) line_div(d_1)

図 7: 信号線分割

る。

5 分割操作による設計可能性

本章では、3章で定義された回路モデルで表される任意の回路が、4章で定義した分割操作によって設計可能であることを以下の手順で示す。

[step1]: 1つのブロックから構成される任意のグラフは構築可能である
[step2]: 2つのブロックから構成される任意のグラフは構築可能である
[step3]: n個のブロックから構成される任意のグラフは2つのブロックで表現することが可能である
[step4]: step1、2、3より、n個のブロックから構成される任意のグラフは構築可能である

以下、各 step について述べる。

[step1]：1つのブロックから構成される任意のグラフは構築可能である

1つのブロックから構成されるグラフには、入力端子からドットを経由、もしくは分岐してブロックに入る経路と、ブロックからドットを経由、もしくは分岐して出力端子に入るか、ブロックへフィードバックする経路しか存在し得ない。

最初に設計を始める段階 (図 8上) でブロック v は存在する。従って、信号線に対して line_div と arc_div をおこな

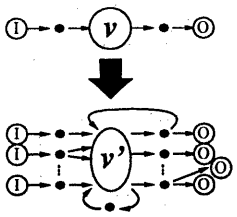


図 8: 一つのブロックから構成されるグラフ

い、入出力端子に para_div をおこなうことによって、任意の入出力線を構築することが出来る。そしてフィードバック線が存在する場合には、loop_add もしくは loop_div を適用することで、1 つのブロックから構成される任意のグラフは構築可能である。

([step1] 終了)

[step2]：2 つのブロックから構成される任意のグラフは構築可能である

まず、2 つのブロック v、w から構成される任意の回路で、接続制約を満足するような全ての信号の流れを考える (図 9)。

今、ドット d に入力しているアークの出力元ノードが v、w、V_{in}、V_{out} のうちのいずれかを返す関数を Node_from(d)、ドット d から出力しているアークの入力先ノードが、いずれかを返す関数を Node_to(d) とする。複数に属する場合は該当する全てを返すものとする。また、$\boldsymbol{P}(V)$ は集合 V のべき集合から空集合 ϕ を引いたものとする。

入力端子からは、ドットを経由してブロック v かブロック w、もしくはその双方に向かう経路が考えられる (図 9(a))。

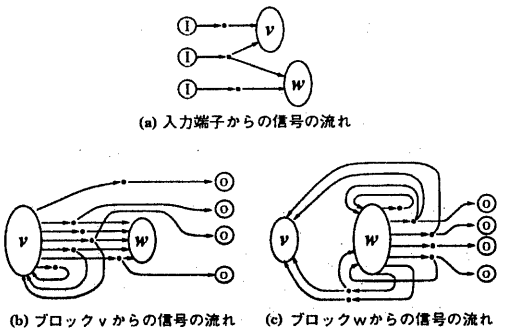
また、ブロック v からは、ドットを経由して出力端子、ブロック w、ブロック v、もしくはそれら複数に向かう経路が考えられる (図 9(b))。

同様にブロック w からはブロック v 同様に、ドットを経由して出力端子、ブロック v、ブロック w、もしくはそれら複数に向かう経路が考えられる (図 9(c))。以上をまとめると、次の式で表される。

Node_from(d) $\subset V_{in}$ ならば
　Node_to(d) $= \boldsymbol{P}(\{v, w\})$
Node_from(d) $= v$ ならば
　Node_to(d) $= \boldsymbol{P}(\{V_{out}, v, w\})$
Node_from(d) $= w$ ならば
　Node_to(d) $= \boldsymbol{P}(\{V_{out}, v, w\})$

これら全種類の信号を用いて接続されている 2 ブロックを、3 種類の分割操作で設計可能であれば、「2 つのブロックから構成される任意のグラフは構成可能である」ことになる。

そこで、まず、図 9に示した経路のうち、フィードバックループが含まれないような単純な 2 ブロック (図 10右下)



(a) 入力端子からの信号の流れ

(b) ブロック v からの信号の流れ　(c) ブロック w からの信号の流れ

図 9: 2 ブロック間の信号の流れ

の設計手順を図 10 に示す。

(1) 入出力端子のドットに対して line_div を、また入出力端子ノードに para_div を適用し、入出力端子を作成
(2) v、w 両ブロックへ分岐させるアークに対して arc_div、ブロック v からブロック w と外部出力とへ分岐させるドットに対して、loop_add を適用し、分岐に必要な信号線を作成
(3) 元のブロックに対して level_div を適用し、入力アークの入力先と出力アークの出力元を指定

図 10の右下の回路には、各種類の信号線は 1 本ずつしか存在しない。しかし、同じ種類の信号線は line_div や arc_div を用いることで増やすことが可能なので、仮に複数本あっても問題はない。

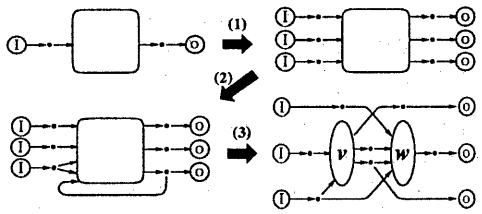


図 10: フィードバックを含まない 2 ブロックの設計例

次に、図 10の右下の回路において、ブロック A とブロック B の間に経路がない場合 (図 11) を考える。前述の level_div を用いてブロックを分割した場合、分割された 2 つのブロックの間に経路が生成されるため、図 11のようなブロックを設計するのには適さない。このような場合には、para_div 、すなわちブロック間に経路を生成しない分割法を用いてブロックを分割することによって設計が可能になる。

以上、フィードバックループを含まない任意の 2 ブロックは、本手法で設計可能であることを示した。

次に、図 9(b)、(c) のような、フィードバックループが含まれる場合について考える。ループ信号は、フィードバック先のノードが自ノードであれば、そのノードを指定

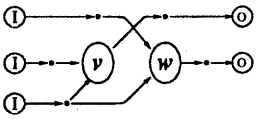


図 11: 2ブロック間に経路のない場合の例

してloop_divを適用するか、フィードバックさせたいアークを指定してloop_addを用いることで、生成することができる。しかし、フィードバック先のノードが一つでない場合には、フィードバック先のノードが分割される前にループ信号を生成する必要がある。

このようなフィードバック信号の設計手順を以下に示す(図12)。

(1) フィードバックさせたい信号経路中のドットに対して、loop_add を適用してループ信号を生成し、生成したアークをフィード先の数だけarc_divによって増加させる。

(2) ブロックに対してlevel_divを用い、その時にループ信号の入力先を指定する。

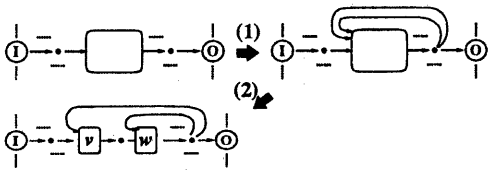


図 12: フィードバックを含む2ブロックの設計例

このように、フィード先のブロックを分割する前にループ信号を生成することによって、複数先にフィードバックする信号を持つような2つのブロックを生成することができる。また、この操作は、フィードバックさせたい信号経路中に位置するドットさえ存在すれば、他の全ての操作とは無関係に行なえる。

これまで述べてきたことから、図9中に表されている信号、すなわち、本モデルによって2ブロックを構成する際に考えられる任意の信号線は、ブロック分割level_div、para_divおよびloop_div、信号線分割line_divおよびarc_divによって設計することが可能であると言える。すなわち、2つのブロックから構成される任意のグラフは構築可能である。

([step2] 終了)

[step3]：n個のブロックから構成される任意のグラフは、接続制約を満たす2個の抽象的なブロックで表すことが可能である

(準備 1) 元のグラフのブロックノード集合全体を V_{block} とし、これを 2つのブロック集合に分けたものを各々 V_A、V_B とする。また、元の回路における全ドットの集合を D_{all} とし、Node_from、Node_toに応じて以下の集合に分類する。

表 1 : D_{all} の分類

集合名	Node_from	Node_to
D_{AA}	V_A	V_A
D_{BB}	V_B	V_B
D_{AB}	V_A	V_B
D_{BA}	V_B	V_A
D_{in}	V_{in}	$P(\{V_A, V_B\})$
D_{o_direct}	V_A or V_B	V_{out}
D_{mul_bra}	上記以外	

上記のドット集合のうち、D_{in}、D_{mul_bra} はそのドットで分岐があるもの、それ以外は分岐がないものである。

(準備 2) 以下に定義する方法によって、上位グラフと下位グラフを構築する。

1. 元のグラフにおいて、V_A、D_{AA} およびそれらの間のアークを単一のブロックAで、V_B、D_{BB} およびそれらの間のアークを単一のブロックBで置き換えたものを、元のグラフに対する上位グラフとする。
2. V_A、D_{AA}、およびそれらの間のアークで構成されるグラフに対して、元のグラフにおいて D_{AA} 以外のドットから V_A に入るアークに対して仮想的な入力端子を、V_A から D_{AA} 以外のドットへ出るアークに対して仮想的な出力端子を加えたものを、ブロックAの下位グラフとする。また、ブロックBの下位グラフも同様に定義する。

nブロックで構成されるグラフの例を図13に、その上位グラフの例を図14に、その時の下位グラフを図15に示す。

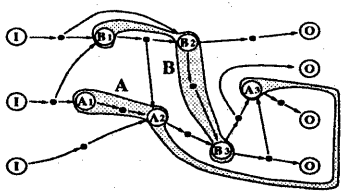


図 13: nブロックで構成されるグラフの例

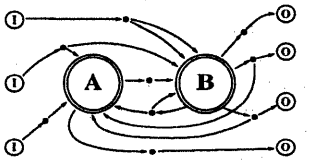


図 14: 図16の上位グラフ

以上のようにして構築した上位グラフは、元のグラフを、2つの抽象的なブロックで表現したものになっている。以下では、上記手順に従って上位グラフと下位グラフ

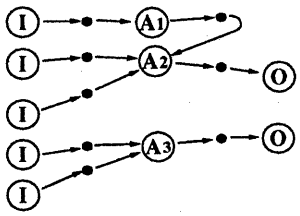


図 15: ブロック A の下位グラフ

を構成した際に、両者が本モデルにおける接続制約を満たしていることを示す。

●上位グラフ

元のグラフにおいて、V_A のノードから D_{AB} のドット d_{AB} を経由して V_B のノードに至る経路、および V_B のノードから D_{BA} のドット d_{BA} を経由して V_A のノードに至る経路は、上位グラフにおいても、入出力している信号線とドットの接続は変化しないので、接続制約を満足する。また、元のグラフにおいて、D_{AA} のドット d_{AA} を経由して V_A のノード間を結ぶ経路、および D_{BB} のドット d_{BB} を経由して V_B のノード間を結ぶ経路は、上位グラフには現れない (以上図 16)。

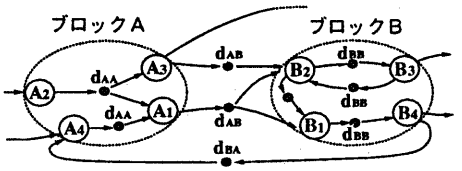


図 16: 上位グラフのノード間における信号線の例 1

入力端子から D_{in} のドットを経由して V_A や V_B のノードに至る経路、および V_A や V_B のノードから D_{o_direct} のドットを経由して出力端子へ至る経路も、上位グラフにおいて信号線とドットの接続は変化しないので、接続制約を満たす。

最後に、V_A や V_B のノードを出発して、D_{mul_bra} のドットで他へ分岐しながら再び同じグループのノードに戻ってくる経路があるが、この経路も、上位グラフを作成した時に、ドットとアークの接続に変化はない (以上図 17)。

以上のことから、上位グラフは、本モデルにおける接続上の制約を満たしていると言える。また、接続制約を満たす任意のグラフから、その上位グラフを構築した際にでき得る信号経路は、自己ループ経路が存在しないこと以外は [step2] で述べた 2 ブロック間における信号経路種類と同じである。

以上、構築した上位グラフは、2 ブロックで構成され、かつ、本モデルにおけるグラフの定義を満たしていることから、[step2] より、本手法で構築可能である。

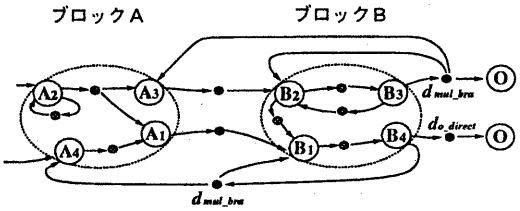


図 17: 上位グラフのノード間における信号線の例 2

●下位グラフ

下位グラフにおいて、ブロック間を結ぶアークとドットの接続関係は、元のグラフと比較して、変更は加えられていない。また、仮想的に加えた入出力端子と、それに付随するアークも接続制約を満足する。したがって、下位グラフが本モデルにおける接続制約を満たしているのは明らかである。

以上により、n 個のブロックから構成される任意のグラフは、手順にしたがって 2 個のブロック、すなわち上位グラフで表すことが可能である。この際、上位グラフと下位グラフは、それぞれ本グラフにおける接続制約を満たす。

([step3] 終了)

[step4]：n 個のブロックから構成される任意のグラフは、構築可能である。

step1、2、3 から、n 個のブロックから構成される任意のグラフは、まず上位グラフを構築後、その各ブロックに対して下位グラフを構築する操作を適用することによって、帰納的に構築可能である。

([step4] 終了)

以上から、本モデルで表現される任意のグラフは、単一ノードに対して 3 章で定義した操作を繰り返し適用することによって、構築することができる。

6 考察

本手法を用いて、実際の回路を設計した結果について述べる。まず、データブック [7] に記載の ALU である LS74381、そして 22 個のインストラクションをもつ 16 ビット計算機 EDC(Elementary Digital Computer)[5] などを設計した。この結果から、本手法の 3 種類の分割操作で、十分に論理回路が設計できることが実証された。図 18に、分割操作で設計した LS74381 を示す。また、表 1 に、本手法で前述の 2 つの回路を設計した時の、回路規模と分割操作数を示す。

LS74381 の設計例では、ゲート数がデータブック記載の回路 (81 ゲート) より 74 % 程度増加している。論理回路の設計品質としては、設計検証、設計変更、再利用に関する設計コスト、ゲート数などの製造コスト、処理速度などの性能の 3 種類が考えられる。本設計手法によって、設計コ

ストが減少するが、回路が冗長になることから、製造コストと性能に関する品質は向上しない。しかし、これらの品質は、本手法によって設計した回路に対して各種最適化を施すことによって、かなり改善されると考えている。

表２： 回路規模と総分割操作数

回路	最小ブロック数	レベル分割	同等分割	ループ分割	総分割操作数
LS74381	141	45	54	0	99
EDC	116	44	36	6	86

一般に、ある回路を設計するための分割操作手順は多数存在する。しかし、回路の理解し易さの観点から、以下の基準を満足する設計手順が望ましいと考えられる。

基準１： フィードバック信号が少ない
基準２： 逐次処理において、入力から直接後段のブロックへ入力する信号線が少ない

実際には、ループ分割の適用時期について十分な考慮が必要になる。２つのブロック間にフィードバック信号が存在する時には、その２つのブロックへの分割を施す前にループ分割を適用しておかなければならない。しかし、設計初期の段階でループ分割を多用すると、前述した基準が満たされなくなる。また、レベル分割の代わりに同等分割を適用してしまうと、ループ分割が余分に必要になる場合があり、基準１が満たされなくなる。

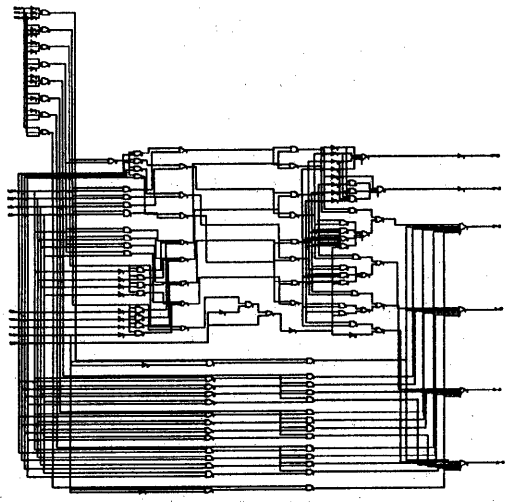


図 18: 分割操作で設計した LS74381

7 まとめ

本稿では、分割操作によって論理回路のトップダウン設計をおこなう際の回路モデル、および操作の定義し、一定の接続制約を満たす論理回路は、３種類の分割操作の組合せによって設計が可能であることを示した。また、実際に本手法を用いて設計を行なった例を示した。本手法によって設計された回路は、接続上ブロック間の機能的関係が明確になる。さらに、本機能的関係に従って、ブロック配置、配線を行い回路図を生成することによって、設計意図をブロックの配置と配線という形で回路図に反映できる[2]。

また、論理合成された回路に対しても、何らかの解析をおこなうことによって、本稿で述べた３種類の関係だけでブロック間の関係を階層的に表現することができれば、回路の表示が理解し易くなり、回路の認識に役立つと考えられる。

今後は、多くの設計例を用いた設計工数および設計品質の評価や、双方向バスの設計に対応するための拡張を考えている。後者を実現するためには、バス上に存在する複数のブロック間の機能的関係をどのようなものに限定すれば良いかという検討が必要になる。

参考文献

[1] M. H.Lang, P. E.Mccormick : “Design Methodologies”, Elservier Science Publishers B.V (North-Holland) pp.123-149(1986)

[2] 小野朗, 新井浩志, 長谷川拓己, 深澤良彰, 門倉敏夫, “トップダウン設計手法に基づいた回路図エディタ : SCHET”, 情報処理学会第 43 回全国大会, 1R-7(1991)

[3] 大和明宏, 新井浩志, 深澤良彰, “論理回路のトップダウン設計における分割操作の適用可能性について”, 情報処理学会第 49 回全国大会, 5L-9(1994)

[4] S.A.Ohr : CAE : A Survey of Standards Trends and Tools, John Wiley and Sons, pp.63-107(1990).

[5] D.L.Dietmeyer : Logic Design of Digital Systems, Allyn and Bacon Inc., pp. 429-473(1979).

[6] R.J.Wilson 著, 斉藤伸自・西関隆夫 共訳, “グラフ理論入門” 近代科学社 (1985)

[7] 74 シリーズ IC 規格表, CQ 出版, p.220 (1992).