システムＬＳＩ設計技術　100－4
（２００１．２．９）

制御処理ハードウェアの高位合成システムにおける面積/遅延見積もり手法

余田 貴幸† 戸川 望‡ 柳澤 政生† 大附 辰夫†

†早稲田大学理工学部電子・情報通信学科
〒169-8555 東京都新宿区大久保 3-4-1
E-mail: yoda@yanagi.comm.waseda.ac.jp
yanagi@yanagi.comm.waseda.ac.jp
to@ohtsuki.comm.waseda.ac.jp

‡早稲田大学理工学総合研究センター
〒169-8555 東京都新宿区大久保 3-4-1
E-mail: togawa@ohtsuki.comm.waseda.ac.jp

本稿では，制御処理ハードウェアの高位合成システムのための面積/遅延見積もり手法を提案する．面積/遅延見積もりでは本システム構成の１つである面積/時間最適化において構築された状態遷移グラフを入力としてその状態遷移グラフに対する面積見積もり値および遅延見積もり値を出力する．提案見積もり手法では状態数および演算器の種類に依存した見積もり式を定式化することでハードウェアの制御部分を含めた面積，遅延の見積もり値を得ている．提案手法をハフマン符号化を始めとするいくつかの制御処理アプリケーションプログラムに適用し，その有効性を評価する．

キーワード　高位合成，動作合成，制御処理，面積/遅延見積もり，コントロールフローグラフ

An Area/Delay Estimation Technique for Control-Based Hardware Synthesis

Takayuki YODA† Nozomu TOGAWA‡ Masao YANAGISAWA† Tatsuo OHTSUKI†

†Dept. of Electronics, Information and Communication Engineering, Waseda University
3-4-1 Okubo, Khinjuku, Tokyo 169-8555, Japan
E-mail: yoda@yanagi.comm.waseda.ac.jp
yanagi@yanagi.comm.waseda.ac.jp
to@ohtsuki.comm.waseda.ac.jp

‡ Advanced Research Institute for Science and Engineering, Waseda University
3-4-1 Okubo, Khinjuku, Tokyo 169-8555, Japan
E-mail: togawa@ohtsuki.comm.waseda.ac.jp

This paper proposes an area/delay estimation technique in high-level synthesis for control flow based hardwares. At area/delay estimation, the input is the state-transition graph, which is generated by the area/time optimizing. The output is estimated area and delay value for the state-transition graph. Our estimation technique gives area and delay including control part of hardware, using an estimation equation. The equation has been decided by number of operations, number of states and type of operations. Experimental results for several control-based hardware demonstrate effectiveness and efficiency of the technique.

Keywords *high-level synthesis, behavioral synthesis, control-based process, area/delay estimation, control-flow-graph*

1 はじめに

一般に，画像符号化・復号化，プロトコル処理といったビット処理あるいは条件分岐処理から構成されるアプリケーションプログラムが専用ハードウェアによって実現されると，ビット処理および条件分岐処理等が並列実行可能となる．このことはマイクロプロセッサによって実現された場合と比較し高速実行可能である．このような制御処理を主体とする専用ハードウェア設計を自動化する高位合成システムは，ビット処理および条件分岐処理といった制御処理を実現するハードウェアを合成することが可能とし，設計者によって与えられた動作仕様に対し複数の設計候補を提供し最適設計を評価する環境を提供すべきと考える．このような考えに基づき，我々は制御処理ハードウェアを対象とした高位合成システムを提案している [7]．本システムは，C 言語によるアプリケーションプログラムの動作記述，アプリケーションデータを入力としてアプリケーションプログラムを実現するハードウェア記述を出力する．このとき，入力されたアプリケーションプログラムに対し時間制約および面積制約を満足するハードウェアを複数個列挙する．

本システムにおける面積/時間最適化系では状態遷移グラフの分割およびコントロールフローグラフの節点に割り当てられている演算器を機能は同じで性能の異なる演算器と交換することで面積および時間の最適化を行なっている [2], [8]．このため面積/時間最適化系では新たに構築した状態遷移グラフに対する評価指標が必要となる．本システムにはハードウェア記述を自動で生成することができるので，見積もりを実行せずに生成されたハードウェア記述より論理合成をしその結果面積値，遅延値を得ることができる．しかし，一般に論理合成には時間がかかってしまうため状態遷移グラフを構築するたびに論理合成を行っていてはハードウェア候補を複数列挙するまでに要する時間が長くなってしまう．このため状態分割や演算器の変更に対する面積や遅延の変化を知るためには面積/遅延見積もりを別に実行する必要がある．高位設計でハードウェアコストを見積もる手法には文献 [1], [3]–[5] などがある．文献 [4], [5] は，面積値や遅延値を AND，OR のゲート数によって見積もられている．文献 [1] は演算器に関するデータベースを使用しスケジューリングやリソースアロケーション，リソースバインディングをほどこした結果から演算器の数を求め見積もり値を得ている．文献 [3] は面積と消費電力を見積もるものであるが，コントロールフローとデータフローを使用して見積もり値を得ている．これらの見積もり手法では演算器のみのハードウェアコストを見積もっており，ハードウェアを制御する制御部分に関しては見積もりをしていない．面積/時間最適化系で実行する状態の分割はハードウェアの制御部を複雑化し，制御部の面積，遅延がハードウェアの面積，遅延に占める割合が大きくなる．このため，演算器のみの面積，遅延の見積もり値だけでなく制御部も含んだ見積もり手法が必要となる．

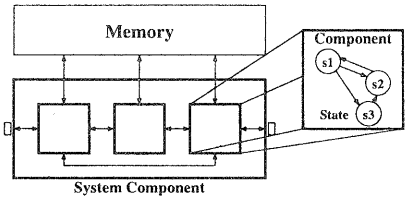

図 1: ハードウェアモデル．

以上の論議より本稿では論理合成を行うことなく面積/時間最適化系にて構築された状態遷移グラフに対する面積および遅延の見積もり値を得る手法を提案する．まず，演算器ライブラリと面積/時間最適化系にて構築された状態遷移グラフを探索，解析を実行した結果得られた演算器の数より面積値および遅延値を求める．性能が異なり同じ機能を持った演算器の交換および状態数の変更に対して制御部の変化を実験より調べることによって状態数および演算器の種類に依存した面積/遅延見積もり式を定式化しハードウェアの制御部を含んだ面積，遅延見積もり値を得ている．

2 諸定義

2.1 ハードウェアモデル

図 1 にハードウェアモデルを示す．ハードウェアモデルは，コンポーネントとメモリから構成される．コンポーネントは，各々独立して動作する状態遷移機械であり，内部にいくつかの状態を持つ．コンポーネントの入出力信号は，外部入出力信号，他のコンポーネントの入出力信号あるいはメモリの内容である．コンポーネント内の各状態では，決められた演算を実行する．コンポーネントの状態は，共通のクロックに同期して遷移する．コンポーネントは，階層的に内部にサブコンポーネントを持つことができる．特に，最上位のコンポーネントをシステムコンポーネントと呼ぶ．メモリは，いくつかのメモリバンクにより構成され，コンポーネントと情報を交換する．異なるメモリバンクに格納された値に対して，コンポーネントは並列に読み出しあるいは書き込みすることができる．本節で定義したハードウェアモデルは，データパスを直接示すものではないが，状態遷移機械の集合として，ビット処理あるいは条件分岐処理といった制御処理を表すことができる．

2.2 コールグラフ，コントロールフローグラフ，状態遷移グラフ

アプリケーションプログラムは，C 言語によって関数の集合として記述される．各関数はハードウェアモデル上のコンポーネントの動作に対応する．各コンポーネントの動作を表す関数を，コンポーネント関数と呼ぶ．

コールグラフ $G_c = (V_c, E_c)$ とは，アプリケーションプログラム全体を表し，アプリケーションプログ

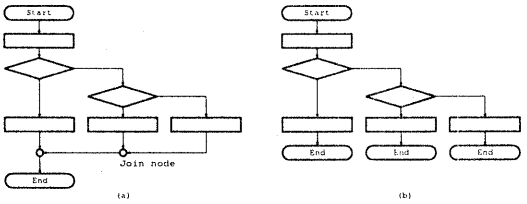

図 2: コントロールフローグラフ (a) と簡易コントロールフローグラフ (b).

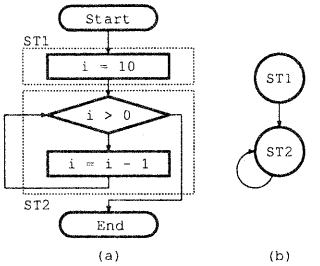

図 3: 簡易コントロールフローグラフ (a) と状態遷移グラフ (b).

ラム中の関数間の呼び出し関係を表す有向グラフである．G_c の各節点 $v \in V_c$ は個々の関数に対応する．関数 v_1 が関数 v_2 を呼ぶとき，G_c は有向枝 (v_1, v_2) を持つ．コールグラフの各節点には，コントロールフローグラフが対応する．

コントロールフローグラフ $G_{cf} = (V_{cf}, E_{cf})$ とは，各関数内部における制御の流れを表すグラフである．特に，関数 f に対応するコントロールフローグラフを明示するとき $G_{cf}(f)$ と書く．G_{cf} の各節点 $v \in V_{cf}$ は，アプリケーションプログラム中の文のうち，次のいずれかを表す．

(a) **始点，終点** コントロールフローグラフの始点あるいは終点を表す．
(b) **演算に対応する節点（演算節点）** 単一の単項演算，2 項演算，比較演算および，ビット結合あるいはビット切り出しといったビット演算 [7] を表す．
(c) **if 文に対応する節点（条件節点）** if 文の条件式を表す．
(d) **switch 文に対応する節点（複数条件節点）** switch 文の条件式を表す．以下本稿で条件節点とは，条件節点と複数条件節点を指すものとする．
(e) **結合節点** 条件節点により分岐した制御の流れを結合する．演算は実行されない．
(f) **while 文に対応する節点（ループ節点）** while 文の条件式を表す．

節点 $v_1 \in V_{cf}$ から節点 $v_2 \in V_{cf}$ への制御の流れが存在するとき，G_{cf} は有向枝 (v_1, v_2) を持つ．

2.3 状態遷移グラフ

コントロールフローグラフ $G_{cf} = (V_{cf}, E_{cf})$ の結合節点 v を考える．v に後続する 1 つの節点 u に対し，u が終点でなければ v と u とを縮退する．u が終点であれば v を破棄し終点を 2 重化する．このような手続きを繰り返して得られるグラフを G_{cf} の**簡易コントロールフローグラフ** (図 2) と呼び，$G'_{cf} = (V'_{cf}, E'_{cf})$ と書く．

今，簡易コントロールフローグラフ $G'_{cf} = (V'_{cf}, E'_{cf})$ から，**状態遷移グラフ** $G_{st} = (V_{st}, E_{st})$ を構築することを考える．G_{st} は，次のように構築される（図 3 参照）．

(1) G_{st} の各節点 $v_i \in V_{st}$ は，始点 s および終点 e を含まない G'_{cf} の連結した部分グラフ $G^i_{cf} = (V^i_{cf}, E^i_{cf})$ に対応する．しかも，V'_{cf} 内の始点および終点を除く全ての節点は，V_{st} 内のいずれかの節点に含まれる．即ち，

$$V'_{cf} = \bigcup_{v_i \in V_{st}} V^i_{cf} \cup \{s, e\}. \quad (1)$$

(2) G_{st} の枝 $e \in E_{st}$ は，次のように構築される．今，G_{st} 内の 2 つの節点 $v_i, v_j \in V_{st}$ に対して，対応する G'_{cf} 内の 2 つの部分グラフを G^i_{cf} および G^j_{cf} とする．このとき，G^i_{cf} から G^j_{cf} に至る枝 $e \in V'_{cf}$ が G'_{cf} 内に存在すれば，G_{st} は枝 (v_i, v_j) を持つ．枝 e が指す G^j_{cf} 内の節点を，節点 v_j における**実行開始点**と呼ぶ．

状態遷移グラフ G_{st} の節点 $v_i \in V_{st}$ が**実現可能**であるとは，v_i に対応する G'_{cf} の部分グラフ G^i_{cf} が，次の条件を満たすときである．

条件 1: G^i_{cf} は，ただ 1 つの実行開始点を持つ

条件 2: G^i_{cf} がサイクルを持たない

v_i が上述の 2 条件を満足するとき，v_i は 1 クロックサイクルで実行可能となり，状態遷移機械の 1 状態として機能する．全ての節点 $v_i \in V_{st}$ が実現可能であるような状態遷移グラフを**実現可能な状態遷移グラフ**と呼ぶ．以降，G_{st} の節点 v_i と状態遷移機械の状態とを同一視する．特に，関数 f に対応する状態遷移グラフを明示するとき $G_{st}(f)$ と書く．$G_{st}(f)$ の有向径路の中で，経由する状態数が最大の径路 P を特定することができる．P を $G_{st}(f)$ の**クリティカルパス**と呼ぶ．P に含まれる状態数を $G_{st}(f)$ の**パス長**と呼び $l(f)$ と書く．

2.4 演算器ライブラリ

演算器ライブラリはコントロールフローグラフ内の演算を実現する演算器の情報を記述したものである．演算器ライブラリは，演算器をハードウェアに組み込むために必要となる．(1) 演算器名，(2) 符号ビット有無，(3) 演算器の実行する演算の種類，(4) 入出力信号の情報，(5) 面積，(6) 遅延，(7) 消費電力を演算器ライブラリに記述する．

設計者は演算器ライブラリの演算器を追加/削除することができる．このため設計者は自身の設計環境に対応した演算器を演算器ライブラリに追加することにより，異なる設計環境に対応したハードウェア記述を生成することが可能となる．面積値，遅延値は演算器を VHDL により記述し論理合成をした結果の値となっている．

3 面積/遅延見積もり手法

制御処理ハードウェアを対象とした高位合成システムでは，面積/遅延最適化の評価値として入力ア

プリケーションプログラムの面積値および実行時間となっている．したがって，面積/遅延最適化系では入力アプリケーションプログラムを実現可能なハードウェアの面積値および遅延値が必要となる．本システムにはハードウェア記述生成系が存在するのでハードウェア記述生成系より出力されたハードウェア記述を論理合成ツールによって論理合成実行することにより面積値，遅延値を得ることができる．一般に論理合成には時間がかかるため，面積/遅延最適化系が最適化をほどこすたびに論理合成ツールを呼び論理合成を実行していてはハードウェア候補を列挙するのに時間がかかってしまう．そのため面積値，遅延値を論理合成を実行せずに見積もる必要がある．

提案見積もり手法の入力は面積/時間最適化系にて構築された状態遷移グラフ，演算器が割り当て済みのコントロールフローグラフ，状態数，および演算器ライブラリである．面積値，遅延値ともにコントロールフローグラフの構造を調べることにより演算器の種類，数を決定し値を得ることができる．面積見積もり値はコントロールフローグラフの構造を調べることにより使用する演算器の種類および数，レジスタの種類および数を決定し，さらに状態数によって値を得ることができる．同様に遅延見積もり値も，コントロールフローグラフの構造より使用する演算器の種類および数を決定し値を得ることができる．

3.1 面積見積もり

面積を見積もるに当たって必要な情報はハードウェアを構成するコンポーネント f に対して対応する状態遷移グラフとコントロールフローグラフの構造，使用される演算器の種類と数，レジスタの数および演算器，レジスタの面積値である．演算器は面積/時間最適化系にてコントロールフローグラフの節点に割り当てられているのでこの数を調べれば f で使用されている演算器の数を決定することができる．しかし，同じ種類の演算器は場合によって共有することができるので演算器数を数えるだけでは見積もり値と論理合成値には誤差が大きくなってしまう．このため，コントロールフローグラフの構造を探索しながら使用されている演算器の数を調べていく必要がある．

3.1.1 演算器の共有

コールグラフ $G_c = (V_c, E_c)$ においてある節点 $v_c \in V_c$ に状態遷移グラフ $G_{st} = (V_{st}, E_{st})$ の節点 $v \in V_{st}$ が複数存在するとき，同種類の演算器が複数の状態節点に存在した場合，この演算器は共有できることがある．図 4 に例を示す．

図 4 の (a) において必要な演算器は加算器と乗算器がそれぞれ 2 つである．(b) の様に状態を分割すると，必要となる演算器は加算器，乗算器それぞれ 1 つずつとなる．これは ST1 で使用した演算器を ST2 で再度使いまわせるからである．一方，(c) の様に分割をすると必要となる演算器は加算器が 1 つ，乗算器が 2 つとなってしまう．

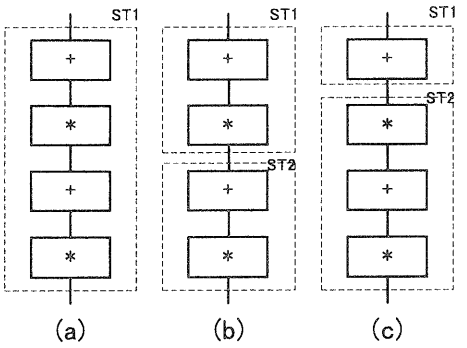

図 4: (a) 状態数が 1 の場合，(b)(a) の状態を 2 つに分割した例 1，(c)(a) の状態を 2 つに分割した例 2

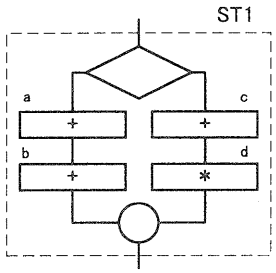

図 5: 条件分岐状態での演算器の共有例

また，同じ状態遷移グラフ節点内においても演算器の共有はある．if 文や swhich 文で表される様な分岐節点があり分岐節点から結合節点までの複数の経路において異なる経路に同種類の演算器が存在した場合これらの演算器は共有することができる．図 5 に例を示す．

図 5 において状態 $ST1$ 中には加算器が 3 つ，乗算器が 1 つが存在している．分岐節点から結合節点までの経路で節点 a を含む経路に分岐した場合，必要となる演算器数は加算器 2 つである．一方，節点 c を含む経路に分岐をした場合，必要となる演算器の数は加算器 1 つ，乗算器 1 つである．条件分岐後はただ 1 つの経路しか実行されないため，用意する演算器の数は分岐節点後から結合節点までの全経路中で必要となる各演算器の最大数となる．この図において用意する演算器の数は加算器 2 つと乗算器 1 つとなる．

この様に状態の分割の仕方，条件分岐後ではいくつかの演算器を共有することができ，コントロールフローグラフを探索していくことにより必要となる演算器の数を得ることができる．

3.1.2 演算器面積の見積もり式

まず，入力となるコントロールフローグラフに対する演算器に関する見積もりを定式化する．前節 3.1.1 で述べたように使用される演算器の数はコントロー

ルフローグラフを探索することにより得ることができる．演算器ライブラリ内にある演算器の集合を HR とし，演算器 $hr \in HR$ の面積値を $a(hr)$ とし，演算器 hr の数を $numalu(hr)$ とする．ある状態 $v_{st} \in V_{st}$ における演算器面積 $A_{alu}(st)$ は

$$A_{alu}(st) = \sum_{hr \in HR} a(hr) \cdot numalu(hr)$$

と表すことができる．実際に論理合成を実行し，演算器面積を求めると演算器を制御する部分が存在するのでこの誤差を α とする．すると上式は

$$A'_{alu}(st) = \alpha \cdot (\sum_{hr \in HR} a(hr) \cdot numalu(hr))$$

と表せる．演算器ライブラリには各演算器に対して遅延に対して最適化をほどこし面積値，遅延値を得たものと面積に対して最適化をほどこし面積値，遅延値を得たものの2種類が存在する．遅延に対して最適化をほどこした演算器の集合を HR_f とし，面積に対して最適化をほどこした演算器の集合を HR_s とする．演算器 $hr_f \in HR_f$，$hr_s \in HR_s$ の面積をそれぞれ $a(hr_f)$，$a(hr_s)$ とし演算器 hr_f と hr_s の数をそれぞれ $numalu(hr_f)$，$numalu(hr_s)$ とするとき，状態 st における演算器面積 $A'_{alu}(st)$ は

$$\begin{aligned} A''_{alu}(st) &= \beta \cdot (\sum_{hr_f \in HR_f} a(hr_f) \cdot numalu(hr_f)) \\ &\quad + \gamma \cdot (\sum_{hr_s \in HR_s} a(hr_s) \cdot numalu(hr_f)) \quad (2) \end{aligned}$$

となる．係数が β と γ の2つになったのは HR_f の演算器のみで論理合成した場合と，HR_s の演算器のみで論理合成をした場合と α の値が異なっていたためである．

式(2)は状態遷移グラフ節点が1つしか存在しない場合である．状態遷移グラフの節点が2以上すなわち状態数 $ST \geq 2$ の場合では演算器の面積見積もりは次のようになる．まず，演算器 hr の数 $numalu(hr)$ は全ての状態のにおいて最も多く使用された数となり $Maxnum(hr)$ と表す．状態 st の演算器 hr の最大数を $num_{st}(hr)$ と表すと，$Maxnum(hr)$ は

$$Maxnum(hr) = \max_{st=1}^{ST} num_{st}(hr) \quad (3)$$

となる．全ての演算器 hr に対して $Maxnum(hr)$ と演算器の面積 $a(hr)$ を乗することにより全ての状態における演算器の面積を得ることができる．複数の状態数が存在するときの論理合成値と式(2)，(3)で得られた見積もり値を比較した結果，見積もり値の方が小さいことがわかった．これは，状態が複数存在するときハードウェアの制御部が複雑化したための誤差であると考える．この誤差の係数を δ，制御部にあたる面積 $A_{control}(st)$ とし，式(2)に適応すると，

$$\begin{aligned} A_{control}(st) &= \delta \cdot (\beta \cdot (\sum_{hr_f \in HR_f} a(hr_f) \cdot numalu(hr_f)) \\ &\quad + \gamma \cdot (\sum_{hr_s \in HR_s} a(hr_s) \cdot numalu(hr_f)) \quad (4) \end{aligned}$$

となり，1状態の演算器を制御する制御部の値となる．この式に全状態数 ST を乗した値が状態数 ST のハードウェアに対する制御部となる．

以上のようコンポーネント f に対する演算器の面積 $A_{alu}(f)$ の見積もり式は

$$\begin{aligned} A_{alu}(f) &= ((\delta \cdot (\beta \cdot A(hr_f) + \gamma \cdot A(hr_s))) \cdot ST) \\ &\quad + (\beta \cdot A(hr_f) + \gamma \cdot A(hr_s)) \quad (5) \end{aligned}$$

$$A(hr_f) = \sum_{hr_f \in HR_f} a(hr_f) \cdot Maxnum(hr_f) \quad (6)$$

$$A(hr_s) = \sum_{hr_s \in HR_s} a(hr_s) \cdot Maxnum(hr_s) \quad (7)$$

$$Maxnum(hr_f) = \max_{st=1}^{ST} num_{st}(hr_f) \quad (8)$$

$$Maxnum(hr_s) = \max_{st=1}^{ST} num_{st}(hr_s) \quad (9)$$

となる．

3.1.3 レジスタ面積の見積もり

本見積もり手法で対象としているレジスタの種類は

- 状態レジスタ
- 入力レジスタ
- 内部レジスタ

の3種類である．これらは本システムのハードウェア記述生成系が作成するレジスタの種類となっている．まず状態レジスタは状態を保持するために必要で各コンポーネントに1つ必要となる．入力信号レジスタは各コンポーネントの入力値を保持するための物でありコンポーネントの入力線の数だけ存在する．内部レジスタは状態が複数ある時，状態を跨いで定義または参照する可能性のある内部変数，およびある状態に置いて参照以前に定義されてない可能性のある内部変数，定義と同時に参照されている内部変数の数だけ存在する．

見積もるレジスタの面積は1ビットのフリップフロップの面積に必要なビット数倍したもので得ることができる．これは design_compiler で論理合成を実行した際，design_compiler は必要なビット幅分の1ビットフリップフロップを用意しているからである．

3.1.4 面積見積もり式

本面積見積もり手法では演算器の面積値を総面積値およびレジスタの総面積値の和によって見積もっている．レジスタの総面積値を $A_{reg}(f)$ で表すと，コンポーネント f に対する面積 $A(f)$ 見積もり式は

$$\begin{aligned} A(f) &= ((\delta \cdot (\beta \cdot A(hr_f) + \gamma \cdot A(hr_s))) \cdot ST) \\ &\quad + (\beta \cdot A(hr_f) + \gamma \cdot A(hr_s)) + A_{reg}(f) \quad (10) \end{aligned}$$

となる．実験より係数 α は 0.0331 となり，β は 1.15，γ は 1.25 となった．

step1. 構築された状態遷移グラフに対して式 (8)(9) を用いて演算器 hr_f と hr_s の最大使用数 $Maxnum(hr_f)$, $Maxnum(hr_s)$ を決定する.

step2. $Maxnum(hr_f)$, $Maxnum(hr_s)$ を用いて式 (6)(7) より演算器の面積値 $A(hr_f)$, $A(hr_s)$ を求める.

step3. $A(hr_f)$, $A(hr_s)$, ST を用いて式 (5) より制御部を含んだ演算器の面積値 $A_{alu}(f)$ を得る.

step4. 状態遷移グラフに対して必要となるレジスタの面積 $A_{reg}(f)$ を求める.

step5. 演算器の面積値 $A_{alu}(f)$ およびレジスタの面積値 $A_{reg}(f)$ を加算することでコンポーネント f に対する面積値 $A(f)$ を求める.

図 6: 面積見積もりアルゴリズム

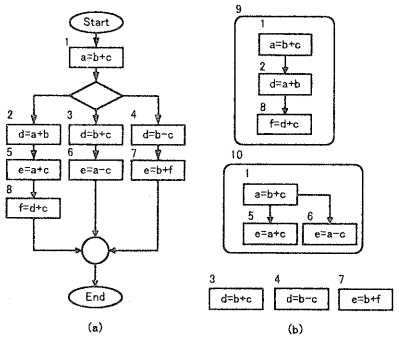

図 7: コントロールフローグラフ (a) と (a) のデータ依存関係 (b)

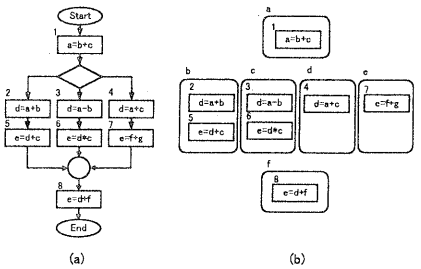

図 8: コントロールフローグラフ (a) と (a) に対する演算器遅延の見積もり方 (b)

3.2 面積見積もりアルゴリズム

前節 3.1 を踏まえて状態数 ST のコンポーネント f に対する面積見積もりアルゴリズムを図 6 に示す.

3.3 遅延見積もり

遅延を見積もる際, 面積と同様にして使用されている演算器の種類と数を探索するだけでは遅延値を見積もることができない. これは評価指標となっている design_compiler がデータ依存の無い演算器同士は同じタイミングで演算ができるように最適化を実行するためである. したがってコントロールフローグラフ上では同一経路上にある演算器もデータ依存が無い場合は全て条件分岐の演算器の共有のようにして考えなくてはならない.

3.3.1 データ依存の考慮

コントロールフローグラフに割り当てられている変数の集合を P とする. 節点 $v_i \in V_{cf}$ 定義されている変数 $p_i \in P$ において参照されている p_i が後段のコントロールフロー節点上に複数個あった場合, これらは全て p_i が定義されている節点 v_i の隣接する後続節点と考えなくてはならない. 節点 v_i で定義されている変数 p_i においてデータ依存がある節点は p_i が後続節点で参照がされている節点 v_j とし定義されている変数を p_j とする. p_j が参照されている後続節点を v_k とし定義されている変数を v_k としたとき, 変数 p_i, p_j, p_k はデータ依存関係にあるので遅延として加算していくべき演算器は節点 v_i, v_j, v_k で使用されている演算器となる. 図 7 に例を示す.

図 7(a) のコントロールフローグラフにおいて節点 1 に着目したとき定義されている変数は a である. a が参照されている節点は節点 2, 5, 6 である. したがってこの 3 つの節点は節点 1 に対して隣接する後続節点となる. 節点 5, 6 で定義されている変数 e は後段の節点では参照されないので節点 1 に対して節点 5, 6 が含まれる節点の遅延は (b) の節点 10 のようになる. 一方, 節点 2 で定義された変数 d は後段の節点 8 で参照されている. したがって節点 1 に対して節点 2, 8 を含む遅延は (b) の節点 9 の様になる. (a) のコントロールフローグラフに対する遅延は (b) の節点 3, 4, 7, 9, 10 の内で最も遅延値の大きい節点となる.

しかし, この方法で遅延を見積もろうとするとコントロールフローグラフ全ての節点に対して変数の参照があるかを調べなくてはならず計算量が大きくなってしまう. そこで程度データ依存関係を調べる範囲を限定して見積もりを行なう. 図 8 に例を示す.

図 8(a) が与えられたコントロールフローグラフとする. 遅延見積もり方としては開始節点から分岐節点もしくは終了節点の間, 分岐節点から分岐節点, 結合節点, 終了節点の間, 結合節点から分岐節点, 結合節点, 終了節点の間の 3 種類を考えてデータ依存を調べる. この方法で (a) のコントロールフローグラフに対する遅延を見積もると

$$d(st) = d(a) + \max_{v \in \{b,c,d,e\}} d(v) + d(f)$$

となる. (a) のコントロールフローグラフのデータ依存関係を考えると, 節点 8 は節点 2 に対してデータ依存関係であるが, 節点 5 には無いので, 上式で考えると節点 5 もしくは節点 8 は過剰に見積もっていることになる. しかし, この方法の方が節点 2 において定義されている変数 d が参照されている節点を調べる節点数は減っている.

3.3.2 演算器遅延の見積もり式

面積見積もりと同様にまず，1状態における演算器遅延の見積もり定式化する．状態 st においてクリティカルパス上にありデータ依存関係のある各演算器の数を $numd_{st}(hr)$ とする．演算器ライブラリより演算器 hr に対する遅延を $d(hr)$ で表すと st における演算器の遅延 $D_{alu}(st)$ は

$$D_{alu}(st) = \sum_{hr \in HR} numd_{st} \cdot d(hr)$$

で求めることができる．面積と同様に制御部分にあたる誤差を論理合成結果のクリティカルパス遅延から求めた．その結果，演算器間の遅延，入力もしくはレジスタから最初の演算器までの遅延，最後の演算器から出力もしくはレジスタまでの遅延と大きく3種類に別けることができた．これらの誤差をそれぞれ σ, φ, ω とし，クリティカルパス上にある全演算器の数を $numd_{cp}(st)$ とすると上式は1状態における各演算器間の制御部を含んだ遅延 $D(st)$ を求める式となり

$$D(st) = \sum_{hr \in HR} numd_{st} \cdot d(hr) + (\sigma \cdot (num_{cp}(st) - 1)) + \varphi + \omega \quad (11)$$

となる．

次に状態数が ST のときの遅延見積もり式を求める．状態数が複数あったとき，遅延見積もり値として出力する値は，最も遅延の大きかった状態 $st \in ST$ の値となり，コンポーネントを動作させる1クロック周期となる．最大遅延となる状態遷移グラフの節点 v_{st} の遅延を求めるために式 (11) は

$$D'(st) = \max_{st=1}^{ST} \{ \sum_{hr \in HR} numd_{st} \cdot d(hr) \} + (\sigma \cdot (num_{cp}(st) - 1)) + \varphi + \omega \quad (12)$$

となる．

3.3.3 遅延見積もり式

面積値を見積もったときと同様に演算器の種類を2種類にし，コンポーネント f に対する遅延 $D(f)$ を見積もる式は式 (12) より

$$D(f) = D(st) + (\sigma \cdot (num_{cp}(st) - 1)) + \varphi + \omega \quad (13)$$

$$D(st)'' = \max_{st=1}^{ST} \{ \sum_{hr_f \in HR_f} (d(hr_f) \cdot numd_{st}(hr_f)) + \sum_{hr_s \in HR_s} (d(hr_s) \cdot numd_{st}(hr_s)) \} \quad (14)$$

となる．ハードウェア記述生成系より出力されたVHDLを実際に論理合成をし，クリティカルパス遅延の値を解析した結果，入力コントロールフローグラフの節点の数によって σ, φ, ω の値が大きく変化していた．このため，入力コントロールフローグラフの節点数 $Node\_no$ によって場合わけを行い，それぞれの値 σ, φ, ω を論理合成結果より求めた．この結果式 (13) は

$$D(f) = \begin{cases} D(st) + (1.32 \cdot (num_{cp}(st) - 1)) \\ \quad + 3.16 + 0.66 \quad (Node\_no > 200) \\ D(st)'' + (1.60 \cdot (num_{cp}(st) - 1)) \\ \quad + 1.16 + 0.52 \quad (else) \end{cases} \quad (15)$$

となった．

step1. 構築された状態遷移グラフに対して状態 st におけるクリティカルパス上にありデータ依存関係のある演算器数 $numd_{st}(hr_f)$ と $numd_{st}(hr_f)$ を決定する．

step2. $numd_{st}(hr_f)$ と $numd_{st}(hr_f)$ の数式 (14) より状態 ST において最大遅延となる遅延値 $D(st)$ を求める．

step3. コントロールフローグラフの節点数 $Node\_no$ より式 (15) を用いてコンポーネント f に対する制御部を含む遅延値 $D(f)$ を求める．

図 9: 遅延見積もりアルゴリズム

表 1: ハフマン符号化器の面積と遅延値

テスト No	面積 (μm^2)		遅延 (ns)	
	論理合成値	見積もり値	論理合成値	見積もり値
1	5757145	5664013	81.57	84.89
2	5218162	4820907	92.80	87.50
3	4608930	4282087	98.06	87.50
4	4104232	4026114	98.06	95.57
5	3923225	3780063	83.46	95.57
6	3831021	3637583	100.98	95.57
7	3756715	3532764	102.93	101.22
8	3468962	3427946	102.93	102.43
9	3327003	3323128	102.93	102.43
10	3197141	3184375	102.79	102.43

3.4 遅延見積もりアルゴリズム

前節 3.3 を踏まえて，コントロールフローグラフの節点数 $Node\_no$，状態数 ST のコンポーネント f に対する遅延見積もりアルゴリズムを図 9 に示す．

4 計算機実験結果

式 (10) および式 (15) を使用したときの見積もり値と論理合成値を以下に示す．論理合成を行ったアプリケーションプログラムはハフマン符号化器，x25 プロトコル [6]，自動販売機コントローラの 3 つである．ハフマン符号化器は decide，encodeac，encodedc，putcode の 4 つのコンポーネントから構成されている．x25 プロトコルは 1 つのコンポーネントから構成されている．自動販売機コントローラは judge_prize，item_proc，coin_hander の 3 つのコンポーネントで構成されている．コントロールフローグラフの節点はそれぞれ 5460，38，360 となっている．本システムの面積/時間最適化系は文献 [2] を使用した．論理合成に使用したツールは，synopsys 社の design_compiler で使用したライブラリは VDEC ライブラリ (0.35μm テクノロジー)[1] を使用し論理合成を実行した．面積と遅延の単位はそれぞれ μm^2，ns である．

計算機実験の結果より面積の論理合成値と見積もり値の誤差は最大で 25%となった．これは論理合成

[1] VDEC ライブラリ (0.35μm テクノロジー) は，東京大学大規模集積システム設計教育研究センターを通し株式会社日立製作所の協力で開発されたものである．

表 2: x25 プロトコルの面積と遅延値

		面積 (μm^2)		遅延 (ns)	
テスト No	状態数	論理合成値	見積もり値	論理合成値	見積もり値
1	5	242208	272201	8.83	7.07
2	6	197944	216111	7.95	7.07
3	6	161139	176011	7.74	8.65
4	7	140112	145399	7.57	8.65
5	7	125064	125823	8.12	8.65
6	8	114388	107846	6.83	8.65
7	8	107816	99281	6.98	10.01

表 3: 自動販売機の面積と遅延値

	面積 (μm^2)		遅延 (ns)	
テスト No	論理合成値	見積もり値	論理合成値	見積もり値
1	484257	461441	18.88	12.94
2	348034	339699	19.34	12.94
3	308811	290837	19.34	13.96
4	291780	270206	19.08	13.96
5	275947	260807	13.81	13.96
6	269853	253593	13.81	13.96
7	263593	246380	13.81	13.96
8	256590	239167	13.81	13.96
9	250331	231955	13.81	13.96
10	238682	227590	13.81	13.96
11	234300	222362	13.81	13.96
12	234403	221987	15.25	12.20
13	221347	203881	15.64	15.56
14	219898	203508	15.67	16.68
15	207728	185402	15.82	16.68
16	201466	176163	15.89	16.68
17	195374	166921	15.89	16.68
18	195772	166786	15.74	16.68
19	188979	171391	12.29	15.56
20	184800	162077	14.35	13.32
21	178340	154331	14.58	13.32
22	175899	151684	14.19	12.20
23	180093	158215	11.98	11.08
24	178473	164745	9.42	11.08
25	172823	150443	9.63	11.08

値，見積もり値ともに値が大きくないため誤差を表しているパーセント表示が大きくなっしまっている．しかし，面積値の平均誤差は約 7%程度となっている．一方，遅延では最大誤差は 43%と大きくなってしまっている．遅延値は面積値と比較して小さいためパーセント表示を行なうと誤差は大きくなってしまう．しかし，遅延値の平均誤差はこちらも 10%程度となった．

論理合成値は面積/時間最適化系をへてハードウェア記述生成系より出力された VHDL を論理合成をして得ている．この出力された全ての VHDL の論理合成値を得るまでにかかる時間は SUN ENTERPISE250(400Mhz*1 メモリ 1G) で x25 プロトコルが 30 分程度，自動販売機では 5 時間程かかり，ハフマン符号化にいたっては 1 つの論理合成値を得るのに 4 時間程かかっている．一方，見積もり値を得る時間は Pentium iii 650Mhz，メモリ 512M で x25 プロトコルで数十秒，自動販売機では 10 分程，ハフマン符号化器では 138 個の見積もり結果を約 15 分程で出力している．これは論理合成を行なった場合と比較して 2000 倍以上高速に面積値および遅延値を得たことになる．この計算機実験の結果より誤差は面積値，遅延値ともに平均 10%以内でで見積もることがでた．また，アルゴリズムが大きいものほど高速に見積もれる結果を得ることができた．

5 おわりに

本稿では制御処理ハードウェアの高位合成システムにおける面積/遅延見積もり手法を提案した．本手法では構築された複数の状態遷移グラフに対して高速に面積遅延見積もり値を得るたことを計算機実験より示した．また，計算機実験の結果よりその有効性を示した．今後は面積および遅延の見積もりだけでなく消費電力の見積もりを可能とすることを目指す．

謝辞

本研究に関し，有用な議論，討論をいただいた，また計算機実験に協力いただいた本学中本真児氏（現松下通信工業），横山正幸氏 (現ソニー)，家長真行氏に感謝いたします．本研究の一部は，文部省科学研究費補助金（基盤研究 C(2)，課題番号 10650345 および奨励研究 (A)，課題番号 12750369）の援助を受けた．

参考文献

[1] J. Gerlach, and W. Rosenstiel, "A scalable methodology for cost estimation in a transformation high-level design space exploration environment," *Proc. of DATE '98*, pp.23–25, 1998.

[2] 家長真行, 戸川望, 柳澤政生, 大附辰夫, "制御ハードウェアの高位合成のための高速な面積/時間最適化アルゴリズム," 情報処理学会 DA シンポジウム'99, pp. 27–32, 2000.

[3] K. S. Khouri, G. Lakshminarayana, and N. K. Jha, "IMPACT: A High-Level Synthesis System for Low Power Control-Flow Intensive Circuits," in *Proc. of Design Automation & Test in Europe Conf.*, pp. 848–854, 1998.

[4] M. Nemani, and F. N. Najm, "Delay estimation of VLSI circuits from a high-level view," in *Proc. DAC'98*, pp. 591–594, 1998.

[5] M. Nemani, and F. N. Najm, "High-level area and power estimation for VLSI circuits," *IEEE Trans. Comput.-Aided Des. Integrated Circuits & Sys.*, pp. 697–713 1999.

[6] A. S. Tanenbaum, *Computer Networks*, Prentice Hall, Englewood Cliffs, N. J., 1989.

[7] 戸川望, 柳澤政生, 大附辰夫, "制御処理を主体としたハードウェアを対象とする高位合成システムとその適用," 情報処理学会 DA シンポジウム'99, pp. 189–194, 1999.

[8] N. Togawa, M. Ienaga, M. Yanagisawa, T. Ohtsuki, "An Area/Time Optimizing Algorithm in High-Level Synthesis for Control Based Hardwares," in *Proc. of ASP-DAC 2000*, pp. 309–312, 2000.