社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2007－SLDM－130 (12)
2007／5／11

消費電力を考慮したprefix graph 合成手法について

松永多苗子† 松永 裕介††

† 早稲田大学大学院情報生産システム研究科 〒808-0135 北九州市若松区ひびきの 2-7
†† 九州大学システム LSI 研究センター 〒814-0001 福岡市早良区百道浜 3-8-33
E-mail: †t_matsunaga@akane.waseda.jp, ††matsunaga@c.csce.kyushu-u.ac.jp

あらまし Prefix graph は parallel prefix adder の概略構造を表現するもので、これまでにそのノード数や段数を加算器の面積や遅延の指標とした合成手法が知られているが、消費電力を考慮した手法の研究はあまり進んでいない。本論文では、テクノロジに依存しないレベルでの消費電力の指標として、ノードの出力のスイッチング確率をとりあげ、タイミング制約下での prefix graph のノード数最小化手法を拡張して、スイッチング確率の総和を最小化する手法を提案する。ノードのスイッチング確率は、そのノードが表すグローバル関数の BDD を用いることによって計算する。また実験を通して、本手法の効果および課題について検討を行なう。

キーワード 演算器合成、parallel prefix adder、 低消費電力、スイッチング確率、動的計画法

On power-conscious approach for prefix graph synthesis

Taeko MATSUNAGA† and Yusuke MATSUNAGA††

† Graduate School of Information, Production and Systems Waseda University 2-7 Hibikino, Wakamatsu-ku, Kitakyushu-shi, Fukuoka 808-0135, JAPAN
†† System LSI Research Center, Kyushu University 3-8-33 Momochihama, Sawara-ku, Fukuoka 814-0001, JAPAN
E-mail: †t_matsunaga@akane.waseda.jp, ††matsunaga@c.csce.kyushu-u.ac.jp

Abstract A prefix graph visualizes a global structure of a parallel prefix adder at technology independent level. Several approaches on prefix graph synthesis targeting area and delay minimization have been proposed so far, but there are few for power. In this paper, switching activity for each node of prefix graph is targeted as one of power measures at technology-independent level. We expand our timing-constrained area minimization algorithm to treat switching activities as the cost to be minimized. Switching activities are calculated by BDD-based method. Effects and issues of our approach are discussed through experimental results.

Key words arithmetic synthesis, parallel prefix adder, low power, switching activitiy, dynamic programming

1. はじめに

加算、乗算等の算術演算回路は、回路全体の品質に影響を与える重要な要素であり、その構成については多くの研究がなされている [3]。その中で、加算器の一種である parallel prefix adder は、桁上げ先見の概念を一般化した手法により桁上げ伝搬を高速化するもので、様々な特徴をもった構造が提案されている [1], [2], [5]。また、構造をあらかじめ固定するのではなく自動生成するアルゴリズムも提案されている [4], [7], [9]。これらの手法は、主としてビットごとに入出力タイミング制約が一様でない場合に、それぞれの状況にあわせて柔軟に構成を決めることでより高速、低面積が実現できるという特徴をもつ。著者らも自動合成手法の一つとして、parallel prefix adder の概略構造を表現する prefix graph に着目し、タイミング制約下での prefix graph 面積最小化手法を提案している [9]。これは、タイミング制約下での面積最小化を目的とした parallel prefix adder 合成問題を、prefix graph 集合に対する探索問題として捉え、動的計画法を利用して体系的に解くもので、既存の手法 [4], [7] に比べて、より大局的に面積最小化を図ることを意図したものである。

一方、回路におけるもう一つの重要な特徴である消費電力に関しては、テクノロジに依存しないレベルである prefix graph において扱う手法はほとんどなく、ようやく、提案されてきつつあるが [11]、まだ確立されたものではない。

本論文では、prefix graph レベルで消費電力を考慮する一つのアプローチとして、消費電力の要因の一つであるスイッチン

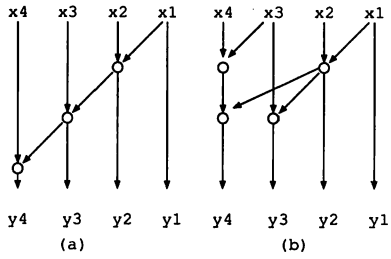

図 1 幅 4 の prefix graph の例
Fig. 1 Examples of prefix graphs: width = 4

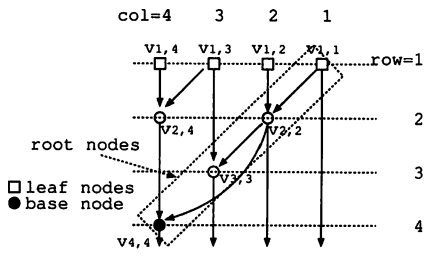

図 2 ノードを入力範囲の幅で整列した prefix graph
Fig. 2 Aligned prefix graph

グ確率に着目し、その最小化手法の提案を行なう。著者らが提案してきた面積最小化手法を拡張して、各ノードのスイッチング確率の総和をコストとして、遅延制約のもとで最小化を行うアプローチを実装し、実験を通して評価を行なう。

以下、準備としてまず prefix adder および prefix graph に関する概念や用語を定義した上で、prefix graph に対する面積最小化問題とその解法について述べる。次に、消費電力を対象として前述の手法を拡張するアプローチについて述べ、実験を通してアプローチの効果、および課題について考察する。

2. 準備: Prefix graph と parallel prefix adder

2.1 Prefix computation と prefix graph

Prefix computation とは、N 個の入力 $x_N, x_{N-1}, \cdots, x_1$ と結合則を満たす任意の演算 $\circ$ が与えられたとき、N 個の出力 y_i $(1 \leqq i \leqq N)$ を、$y_i = x_i \circ x_{i-1} \circ \cdots \circ x_1$ によって計算するものである。これは、i 番目の出力 y_i は、$j \leqq i$ となる入力 x_j のみに依存することを意味する。

[定義 1] N 入力 prefix graph は、N 個の入力に対する prefix computation における各演算 $\circ$ をノードとした非巡回有向グラフ (directed ascyclic graph: DAG) であり、各出力の値を計算するための演算の実行順序を表現したものである。ノードのファンインは 2 項演算 $\circ$ の 2 つのオペランドに対応し、ノード v_1 が v_2 のオペランドであるとき、v_1 から v_2 へのエッジが存在する。v_1 を v_2 のファンインノード、v_2 を v_1 のファンアウトノードと呼ぶ。Prefix graph の入力数 (および出力数)N を、prefix graph の幅と呼ぶ。

図 1 に幅 4 の prefix graph の例を示す。(a) の y_4 は、$y_4 = x_4 \circ (x_3 \circ (x_2 \circ x_1))$ という演算順で計算され、(b) では、$y_4 = (x_4 \circ x_3) \circ (x_2 \circ x_1)$ で計算される。Prefix computation は結合則が成り立つため、どちらの順序で計算しても結果は変わらない。

図 2 は、図 1(b) の各ノードを入力範囲の幅で整列したものである。行 i、列 j のノード $v_{i,j}$ $(i \leqq j)$ は、i 個の入力、$x_{j-i+1}, \cdots, x_j$ に対する prefix computation の中間結果を表わしている。この入力範囲を $[j : j-i+1]$ と記す。演算の 2 つのオペランドは、連続した入力範囲に対する中間結果となっている。

例えば、図 2 の $v_{2,4}$ は 2 つのファンイン $v_{1,4}, v_{1,3}$ を持ち、$v_{2,4} = v_{1,4} \circ v_{1,3}$ という演算結果を表している。$v_{2,4}$ は入力 3 から 4 に対する演算結果、$v_{2,2}$ は入力 1 から 2 に対する演算結果であり、それらに演算を施した結果 $v_{4,4}$ は入力 1 から 4 に対する演算結果を表わしている。

[定義 2] 幅 N の prefix graph において、行 1 上のノード $v_{1,j}$ をリーフノードと呼び、$v_{j,j}, 1 \leqq j$ をルートノードと呼ぶ。リーフノードは入力、ルートノードは出力に対応している。ルートノードの中で最大の幅をもつノードをベースノードと呼ぶ。

2.2 Prefix computation としての 2 進加算

n ビットの 2 進加算の入力を $A = a_n, \cdots, a_1, B = b_n, \cdots, b_1$, 出力を和 $S = s_n, \cdots, s_1$, 及び、キャリー出力 c_n とすると、各ビットの和 s_i とキャリー出力 c_i は、$s_i = a_i \oplus b_i \oplus c_{i-1}$, $c_i = a_i b_i + a_i c_{i-1} + b_i c_{i-1}$ で計算される。この n ビット加算は、個々のビットに対する桁上げ生成/伝搬関数 (g, p)、それを拡張した複数ビットのグループに対する桁上げ生成/伝搬関数 (G, P) を用いると以下のように計算できる。

- g,p の生成: $g_i = a_i \cdot b_i$, $p_i = a_i \oplus b_i$
- prefix 処理: G, P を用いて $c_i = G_{[i:1]}$ を計算

$$G_{[i:j]} = \begin{cases} g_i & \text{if } i = j \\ G_{[i:k]} + P_{[i:k]} \cdot G_{[k-1:j]} & \text{otherwise} \end{cases}$$

$$P_{[i:j]} = \begin{cases} p_i & \text{if } i = j \\ P_{[i:k]} \cdot P_{[k-1:j]} & \text{otherwise} \end{cases}$$

- s_i の生成: $c_i = G_{[i:1]}, s_i = p_i \oplus c_{i-1}$

Parallel prefix adder は上記の 3 つの処理を行う要素から構成される加算器である。このうち、prefix 処理の部分は、演算 $\circ$ を以下のように定義することによって、prefix computation とみなせ、その演算順序は prefix graph で表現することができる。

$$\begin{aligned} (G,P)_{[i:j]} &= (G,P)_{[i:k]} \circ (G,P)_{[k-1:j]} \\ &= (G_{[i:k]} + P_{[i:k]} \cdot G_{[k-1:j]}, \\ &\quad P_{[i:k]} \cdot P_{[k-1:j]}) \end{aligned}$$

2.3 Parallel prefix adder の概略構造としての prefix graph

Parallel prefix adder の概略構造は prefix 処理部の構成に依存し、その構成は prefix graph によって表現される。加算器の

面積は、prefix graph のリーフ以外のノード数で測るものとし、prefix graph のサイズと呼ぶ。遅延に関しては、prefix graph の各ノード v に対して、入力からの遅延時間に相当する到達レベル $AL(v)$ と、そのノードの到達レベルが満たすべき要求レベル $RL(v)$ を以下のように定義する。

$$AL(v) = max\{AL(v'),\ v' \in FI(v)\} + 1$$

$$RL(v) = min\{min\{RL(v'), v' \in FO(v)\} - 1,\ RL'(v)\}$$

ここで、$FI(v)$ は v のファンインノード、$FO(v)$ は v のファンアウトノード、$RL'(v)$ は自分自身に与えられた要求レベル (未定の場合は ∞ とする) を示す。

例えば、図 3 において、ビットごとの出力要求レベルが $RL(v_{1,1}) = 5, RL(v_{2,2}) = 5, RL(v_{3,3}) = 5$, および $RL(v_{4,4}) = 5$ であるとする。このとき、$RL(v_{2,4})$ は 4 であり、$RL(v_{2,2})$ は再計算され 4 になる。

入力ノードに対する到達レベル、出力ノードに対する要求レベルは、遅延制約として外部から与えられるものとする。ある prefix graph が制約を満たすとは、そのすべてのノード v において、$RL(v) \geqq AL(v)$ が成立することである。

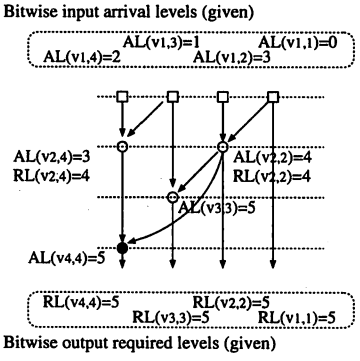

図 3 到達レベルと要求レベル
Fig. 3 Arrival levels and required levels

3. Prefix graph に対する面積最小化手法

本節では、[9] で著者らが提案した prefix graph の面積最小化手法について説明する。本手法は、タイミング制約下での面積最小化問題を、与えられたビットごとの入力到達レベル、出力要求レベルの下での、サイズ最小となる prefix graph 生成問題としてとらえるものである。

Prefix graph は DAG であり、実用的なビット幅の prefix graph に対して遅延制約下での面積最小化の厳密解を効率よく得るのは困難である。そのため、既存の手法では、まず各ノードの到達レベルが最小になるような構成を求めてから、タイミング制約に余裕がある範囲で、面積を削減するための局所的な構造変換を行う、という手法が用いられている [4], [7]。これに対して本手法では、面積最小化に対してより大局的な視点で取り組むことを目指して、2 つのプロセスからなるアルゴリズムを提案する。まず、最初のプロセス、動的計画法に基づく面積最小化 (Dynamic Programming based Area Minimization: DPAM) では、prefix graph のある部分集合に集中して最小解の探索をする。この部分集合を以下では制限付 prefix graph と呼ぶ。探索する部分集合を適切に定義することにより、その部分集合に対して効率よく動的計画法を適用することが可能になり、厳密最小解が得られる。ただしこの厳密性は部分集合に対するもので、prefix graph 全空間を対象とした場合には、まだ面積削減の余地が残っている。そこで、次のプロセス、再構築による面積削減 (Area Reduction with Re-Structuring:ARRS) において、構造に対して与えた制限を取り外してより小さな prefix graph となるよう再構築する。

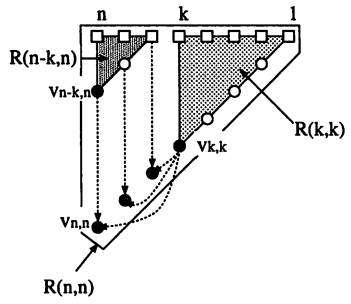

図 4 制限付 prefix graph
Fig. 4 Restricted prefix graph

本手法のキーとなるのは制限付き prefix graph は、以下のように定義する。

[定義 3] 幅 n の prefix graph のベースノード $v_{n,n}$ の 2 つのファンインノードを、$v_{n-k,n}$, $v_{k,k}$ とする。制限付 prefix graph $R(n,n)$ とは、その構造が次の特徴をもつ prefix graph の集合である (図 4):

- 各ルートノード $v_{n-j,n-j}$ $(0 < j < n-k)$ のファンインノードは、$v_{k,k}$, $v_{n-k-j,n-j}$ である。
- $v_{n-k,n}$, $v_{k,k}$ をそれぞれベースノードとする部分グラフもまた、同様の特徴を再帰的にもつ。

部分グラフに含まれないノード $v_{n-j,n-j}$, $0 \leqq j \leqq n-k$ を $R(n,n)$ の固有ノードと呼ぶ。

制限付 prefix graph には、極端な場合としてシリアルな prefix graph (リップルキャリーに相当) や Sklansky 型の prefix graph が含まれる。

3.1 動的計画法の基づく面積最小化 (DPAM)

DPAM では、制限付き prefix graph を対象として、与えられた入力到達レベル、および、出力要求レベルの下で、ノード数最小化を厳密に解く。

制限付 prefix graph は互いに排他的な部分グラフを組合せた構造であり、その部分グラフも再帰的に同様の構造をもつ。その結果、制限付 prefix graph の最小化問題は最適性の原理が成り立ち、部分問題の最適解を組み合わせて全体の最適解を導くことが可能となる。例えば、$R(n,n)$ の場合、ベースノードのファンインノード対の候補は、

$\{v_{1,n}, v_{n-1,n-1}\}, \{v_{2,n}, v_{n-2,n-2}\}, \cdots, \{v_{n-1,n}, v_{1,1}\}$ の $n-1$ 通りがあり、各ケースにおける最小サイズは、そのファンインノードをベースとする2つの部分グラフに対する部分問題の最適解に基づいて計算できる。$n-1$ 通りの中でサイズが最小値となる構成が $R(n,n)$ に対する最適解となる。したがって、サイズの小さい順に部分問題を解き、それらの結果を用いることで全体の解が求められることになる。

部分問題を解く際、部分グラフが満たすべき出力要求レベルは、その部分グラフを構成要素とする元のグラフから一意に決まる。しかし、サイズの小さい部分問題から順に処理していく時点では、その推移的ファンアウトの構成は確定しないため、出力要求レベルを決定できないため問題が解けない、という問題がある。これに対して、本手法では、テクノロジマッピング技術で用いられているのと同様に、各出力ノードに想定される要求レベルの値の組合せごとに、最適解を保持する、というストラテジを用いる [6]。すべての部分問題を処理すると要求レベルは決定されるので、決定した値の組合せに対応した最適解を選択することで、全体を構築することができる。

要求レベルが設定されるノードの数は、幅 n' の部分グラフに対して n' 個存在し、それぞれが複数の値を取りうるため、その全組合わせを保持して扱うのは現実的ではない。しかし、制限付き prefix graph の構造の特徴を用いると、厳密性を失うことなく、各部分問題に対して n' ではなく、2個の値のみ保持することで計算可能となるとともに、その値が取りうる範囲も制限することができるため、結果的に実用的な時間での処理が可能となっている。

3.2 Prefix graph の再構築による面積削減 (ARRS)

ARRS は、DPAM の結果得られた prefix graph を出発点として、構造上の制限を取り除いてグラフを再構築することにより、面積のさらなる削減を図るものである。

制限付き prefix graph において、そのグラフの固有ノードのすべての斜めファンインノードは共有される。すなわち、ベースノード $v_{n,n}$ のファンインノードが $v_{n-k,n}$ と $v_{k,k}$ であるような制限付き prefix graph において、すべての固有ノード $v_{n-j,n-j}(j < n-k)$ は共通のファンインノード $v_{k,k}$ をもつ (図 5(a))。この制約を取り払ってファンインの付け替えを行なう場合、ある固有ノード $v_{n-j',n-j'}$ の新しいファンインノードの位置としては、以下の2つのケースが考えられる：

- ケース 1 : $n-j' > k$ (図 5(b))

あるノード $v_{r0,c0}$ の2つのファンインノード $v_{r1,c1}$、$v_{r2,c2}$ の行数の間には、$r0 = r1 + r2$ という関係が成り立っている。したがって、ケース1は、新しい垂直ファンインノードの行数はもとの垂直ファンインノードの行数より小さくなることを意味している。この場合、新しい垂直ファンインノードは $v_{n-k,n}$ をベースとする部分グラフを構成するノードとして存在している可能性があり、その場合には、この再構築によって新たにノードが必要となることはない。さらに、もし $v_{n-j,n-j}$ が $v_{n-k-j,n-j}$ の唯一のファンアウトノードであるならば、再構築により $v_{n-k-j,n-j}$ は出力に到達不能になるため、削除することが可能となり、さらにその推移的ファンインノードに対しても同様の状況下でノードの削減が可能となりうる。

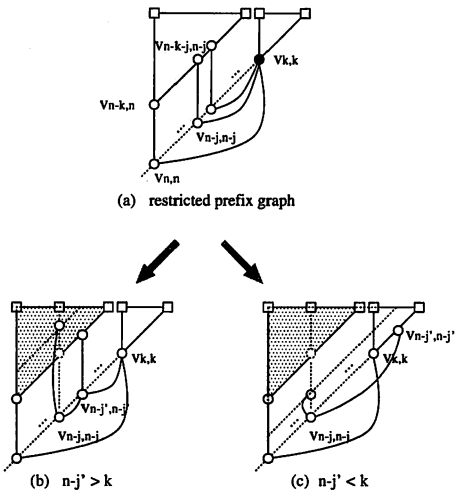

図 5 制限付き prefix graph を起点とした再構築
Fig. 5 Restructuring of a restricted prefix graph

- ケース 2 : $n-j' < k$ (図 5(c))

ケース1の場合と対照的に、このケースにおいては垂直ファンインノードの行数はもとの行数より大きくなり、このことは、新しいファンインノードは既存のサブグラフには含まれないことを意味する。したがって、もしこのような再構築を実行すると、新しいファンインノードをベースとする prefix graph を構築する必要があり、ノード数が増加することになる。

ARRS では、新しいノードを導入することなくノードを削除するという方針をとり、上述のケース1のみを対象とする。付け替えの方法としては、ケース1に属する任意の付け替えを対象とした下記の手法を採用している：

- 各ノード $v_{n-j,n-j}$ に対して、タイミング制約を満たす中で可能な付け替え候補に対して、付け替えによるゲインを計算する。ゲインは、その斜めファンインノードを当該候補に変更した場合に削除できるノード数である。
- ゲイン最大の変更を採用してファンインを付け替える。それによりタイミングの再計算を行なう。
- ゲインが得られなくなるまで繰り返し、ゲインが得られなくなったら、部分グラフ $v_{n-k,n}$, $v_{k,k}$ をベースとする部分グラフそれぞれについて同様の面積削減を繰り返す。

4. 消費電力を考慮した prefix graph 合成

4.1 prefix graph における消費電力の指標

Prefix graph はテクノロジに依存しないレベルでの parallel prefix adder の概略構造を表現するものであるため、対象テクノロジや、マッピング、配置配線等の後続処理に対する何らかの仮定をおかずに詳細な指標を考えることは困難である。本論文では、このレベルで考えられる消費電力要素として、ノードのスイッチング確率に基づいた指標を取り上げる。

回路の動的な消費電力は、各ゲートを駆動する容量 C_i, 電源

電圧 V, スイッチング周波数 f_i とすると $\Sigma C_i V^2 f_i$ に比例する。この中で、f_i を削減することのみに着目して、prefix graph 上のノードに対してスイッチングコスト SW を定義する。

まず、仮定として加算器への各入力信号 i に対して、その信号が 1 になる確率 $P(i)$(1 出現確率とよぶ) が与えられ、かつ、ビットごとに依存性がないとする。各 minterm の出現確率は、各ビットの出現確率の積で求められ、さらに、関数 f に対する出現確率は、そのオンセットに含まれる minterm の出現確率の総和を求めることで得られる。

ある信号 f が 1 から 0, あるいは、0 から 1 に遷移する確率 (以下、スイッチング確率とよぶ) SW(f) は、その 1 出現確率を P(f) とすると以下の式で計算される:

$$\begin{aligned} SW(f) &= (P(f)\cdot(1-P(f)) + (1-P(f))\cdot P(f)) \\ &= 2\cdot P(f)\cdot(1-P(f)) \end{aligned}$$

Prefix graph のノードは、2.2 節で定義した式で与えられた論理を表している、gi, pi の 2 種類の出力をもつ。prefix graph のノードのスイッチングコスト $SW(i)$ を以下のように定義する:

$$SW(i) =. \alpha\cdot SW_g(i) + \beta\cdot SW_p(i)$$

$$SW_g(i) = 2P(gi)\cdot(1-P(gi))$$

$$SW_p(i) = 2P(pi)\cdot(1-P(gi))$$

4.2 ノード出力の 1 出現確率の計算

1 出現確率の計算には、二分決定グラフ BDD [8] を用いたアプローチを採用する [12]。ノードのある出力のグローバル関数を f とし、Shanon 展開すると $f = xf_x + x'f_{x'}$ となる。この関数の 1 出現確率は、

$$\begin{aligned} P(f) &= p(x)p(f_x) + p(x')p(f_{x'}) \quad (1) \\ &= p(x)p(f_x) + (1-p(x))p(f_{x'}) \quad (2) \end{aligned}$$

したがって、f のグローバル関数を表す BDD、および、各変数の 1 出現確率が与えられれば、BDD をたどりながら P(f) を計算することができる。

4.3 アプローチ

SW コストは各ノードの SW コストの総和であり、制限付き prefix graph を対象とした場合に、互いに排他的な部分グラフのコストの総和として計算できるため、コストを SW に置き換えてノード数の場合と同様の枠組で解くことが可能である。

(1) DPAM フェイズ:
コストとして各ノードの SW コストを用いる。ノード数を用いる場合との違いは、SW コストはノードごとに異なるため、それぞれのコストを計算する必要がある。例えば、図 4 において、ノード数の場合、部分グラフが決まれば、そのときの固有ノード数はその構造から自動的に決まる。しかし、SW の場合は各固有ノードに対して SW を計算する必要がある。これに対して、現在のところ想定される要求レベルの各ケースに対して最適解を保持する際に、固有ノードの BDD を保持して、よりサイズの大きな問題を解く際には、それを用いて固有ノードの BDD を計算するようにしている。

(2) ARRS フェイズ:
再構築の手続き自体は同様の枠組で行ない、ゲインの計算部分を SW コストのの減少分に置き換える。

5. 実験と考察

SW をコストとすることの効果を評価するために上述の手法を実装し、下記のタイミング制約下でスイッチングコストの最小化を対象とした prefix graph 合成の実験を行ない、面積最小化を対象として合成した場合と、SW コストの比較を行なった。

- 24-bit 加算器:

入力到達レベルは一様 (0)、出力要求レベルは一様で最小レベル数 (5)

- 24-bit 加算器 (Wallace tree 型 16bit 乗算器の最終段加算器):

入力到達レベルは、乗算器の部分積計算部分の構成から生成 ([10])、出力要求レベルは一様で最小レベル数

- 24-bit 加算器 (Wallace tree 型 16bit 乗算器の最終段加算器):

上記と同様で、出力レベルは最小レベル数+1

- 32-bit 加算器:

入力到達レベルは、中間ビットが遅くなる凸型、出力要求レベルは一様で最小レベル数

実験においては、SW コストのファクタ α, β はともに 1 として計算した。

表 1 は、入力変数の 1 出現確率が均一 (0.5) であった場合の結果を示している。表において、dpam, dpam+arrs, dppm, dppm+prrs の列は、面積を対象とした場合のフェイズ 1、フェイズ 1 と 2、スイッチングコストを対象とした場合のフェイズ 1、および、フェイズ 1 と 2 の各処理により得られた prefix graph のコスト (同一データに対する 1 行目はスイッチング確率の総和、2 行目はノード数の総和) を示している。表からわかるように、これらのケースにおいて、スイッチング確率をコストとして用いる優位性は、特にビット幅が比較的小さい場合においてみられなかった。1 出現確率が均一で 0.5(1,0 同確率でおこる) 場合、 各ノードのスイッチングコストはほぼ一定で、各ノードあたり 0.5 前後の値であり、ノード間の差は 0.1 くらいの幅しかなかった。差がでたのは、動的計画法を用いて最小解を生成する際、ノード数としては等価であるがスイッチングコストとしては微小な差がつく候補が存在した場合であった。コストが同じであればどちらを選んでも制限付き prefix graph に対する最小解は保証されるが、そこで決めた構造により再構築の処理に影響がでる場合がある。スイッチングコストの差によって選ばれた候補と異なる候補が面積最小解では選ばれていた場合、DPAM 終了時で生成される概略構造の骨格が変わる場合は起こりうる。今回の例では、むしろ再構築による削減効果が減ってしまい、結果的にコストが大きくなる場合があった。このことは逆に考えると、入力変数の 1 出現確率を 0.5 と仮定する場合は、ノード数を最小化することで、同時にスイッチング確率の総和も下げていることになり、より複雑なスイッチングコストを対象とする必要はないとも考えられる。

表 1 実験結果: 入力の 1 出現確率が一定 (0.5) の場合

	dpam	dpam+arrs	dppm	dppm+prrs
24min	44.6066	42.7509	44.3610	43.4602
	49	45	49	47
p24	44.6401	44.1862	44.4879	44.4879
	49	50	50	50
p24-1	40.1758	40.1758	40.1669	40.1669
	41	41	41	41
p32	54.7114	54.2294	54.5588	54.0768
	57	56	57	56

表 2 入力の 1 出現確率がビットごとに異なる場合の一例

	dpam	dpam+arrs	dppm	dppm+prrs
24min	39.4358	37.8743	38.9283	37.7614
	49	45	49	46
p24	39.4380	39.0113	39.3758	38.9491
	49	50	50	50
p24-1	35.7642	35.7642	35.7642	35.7642
	41	41	41	41
p32	48.3228	47.9060	47.5062	47.0895
	57	56	57	56

表 3 処理時間-1 出現確率が 0.5 の場合 (秒) (Pentium 4, 2.4GHz)

	Amin(uni)	Pmin(uni)
24min	1.97	2518.78
p24	2.09	1478.45
p24-1	1.95	5093.53
p32	2.44	11068.96

表 4 処理時間-1 出現確率がランダムな場合 (秒) (Pentium 4, 2.4GHz)

	Amin(rand)	Pmin(rand)
24min	1.97	2517.50
p24	1.99	1486.07
p24-1	1.72	5088.40
p32	2.42	11104.93

一方、入力変数の 1 出現確率は一様であるとは限らず、その場合には、各ノードのスイッチング確率には差が生じると考えられる。その一例として、入力ビットごとの出現確率をランダムに与えた場合について実験を行った。表 2 に示すように、この場合にはすべてにおいて、スイッチング確率をコストとした手法の方が、小さい結果が得られた。ノード数で比較した場合には大きくなる場合もあり、スイッチング確率を直接最小化の対象とすることの優位性を示している。ただし、表 3, 4 に示すように、32bit 程度で数時間かかっており、現状ではそれ以上サイズが大きい場合、実用的ではない状況である。SW 計算のためのアルゴリズム、および、実装上の改良が課題である。

6. おわりに

本論文では、prefix graph に対してスイッチング確率のコストを導入し、タイミング制約の下でのノード数最小化手法の枠組みを利用して、スイッチング確率の総和を最小化する手法について提示し、実験を通して評価、考察を行った。その結果、入力出現確率が一定 (0.5) の場合には、スイッチング確率を考慮する優位性は必要はほとんどないが、出現確率がばらつく場合には、面積最小構成のスイッチング確率と比較してより小さい結果が得られ、prefix graph レベルでの有効性が確認された。ただし、現状ではサイズに問題があるため、アルゴリズムおよび実装上の改良が課題である。また今回の手法は、スイッチング確率単独を対象とし、その値でのみの比較であるため、実回路上での消費電力削減における効果を調べる必要がある。

7. 謝　辞

本研究の一部は、福岡地域の文部科学省知的クラスター創成事業の支援による研究成果を基盤としている。

文　献

[1] R. P. Brent and H. T. Kung. A regular layout for parallel adders. *IEEE Trans. Computers*, 31(3):260–264, March 1982.

[2] P. M. Kogge and H. S. Stone. A parallel algorithm for the efficient solution of a general class of recurrence equations. *IEEE Trans. Computers*, 22(8):786–793, August 1973.

[3] I. Koren. *Computer Arithmetic Algorithms*. A K Peters, Ltd., 2002.

[4] J. Liu, S. Zhou, H. Zhu, and C.-K. Cheng. An algorithmic approach for generic parallel adders. In *ICCAD*, pages 734–730, November 2003.

[5] J. Sklansky. Conditional sum addition logic. *IRE Trans. Electron. Comput.*, 9(6):226–231, 1960.

[6] H. Touati, C. Moon, R. K. Brayton, and A.Wang. Performance-oriented technology mapping. In *MIT VLSI Conference*, 1990.

[7] R. Zimmermann. Non-heuristic optimization and synthesis of parallel-prefix adders. In *International Workshop on Logic and Architecture Synthesis*, pages 123–132, December 1996.

[8] R. Bryant. "Graph-based algorithms for Boolean function manipulation", In IEEE Trans. Comput. C-35,8(Aug.), pp.677–691, 1986.

[9] Taeko Matsunaga and Yusuke Matsunaga, "Area Minimization Algorithm for Parallel Prefix Adders under Bitwise Delay Constraints", in ACM Great Lakes Symposium on VLSI(GLSVLSI'07), March 2007.

[10] 松永多苗子、松永裕介、"Parallel prefix adder 合成を用いた乗算器の最適化手法について", 第 20 回 回路とシステム軽井沢ワークショップ, 2007 年 4 月

[11] J. Liu, Y. Zhu, H. Zhu, C-K. Cheng and J. Lillis, "Optimum Prefix Adders in a Comprehensive Area, Timing and Power Design Space", in ASPDAC 07, pp.609–615, January 2007.

[12] M. Pedram, "Power Minimization in IC Design: Principles and Applications", in ACM Trans. on Design Automation of Electronic Systems, Vol.1, No.1, January 1996, pp.3–56