Ullmann のアルゴリズムのハードウェアによる実装に関する研究

指導教官 市川周一　　学籍番号 963715 齋藤 秀充

1　はじめに

Ullmann のアルゴリズム [1] は，代表的な部分グラフ同型判定アルゴリズムの 1 つである．本研究では Ullmann のアルゴリズムを様々な方法でハードウェアに実装した場合の，回路規模と実行速度について比較検討する．

2　Ullmann の実装方法

グラフ $G_\alpha = (V_\alpha, E_\alpha), G_\beta = (V_\beta, E_\beta)$ を考える．V_α, V_β はそれぞれ G_α, G_β の頂点集合，E_α, E_β はそれぞれ G_α, G_β の辺集合である．$p_\alpha = |V_\alpha|, p_\beta = |V_\beta|$，隣接行列を $A = [a_{ij}]\ (1 \le i,j \le p_\alpha)$, $B = [b_{ij}]\ (1 \le i,j \le p_\beta)$ とする．$v_{\alpha_i} \in V_\alpha$ から $v_{\beta_j} \in V_\beta$ への写像で部分グラフ同型になる可能性がある場合に $m_{ij} = 1$，なければ $m_{ij} = 0$ として，行列 $M = [m_{ij}](1 \le i \le p_\alpha, 1 \le j \le p_\beta)$ を定義する．

Ullmann[1] は，以下の式を再帰的に適用して m_{ij} を計算することにより，同型可能性を判定する方法を示した (Refinement Procedure)．$r_{xj}(1 \le x \le p_\alpha, 1 \le j \le p_\beta)$ は中間変数である．

$$r_{xj} = (\exists y)(m_{xy} \cdot b_{yj}) \tag{1}$$

$$m_{ij} = m_{ij} \cdot (\forall x)(\bar{a}_{ix} \vee r_{xj}) \tag{2}$$

m_{ij} の修正は i, j の順序に関係なく行うことができるので，図 1の回路 (以下，sub_comb と呼ぶ) を $p_\alpha p_\beta$ 個並べることにより Refinement Procedure を実装することができる．以下この方法を **comb** と呼ぶ．

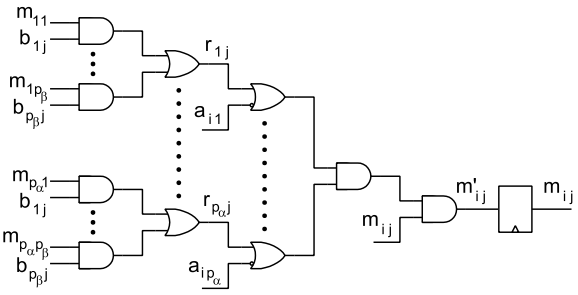

図 1: Refinement Procedure を実現する組合せ回路 [1]

3　順序回路による実装

Ullmann の提案 (組合せ回路) による実装は，実行時間が短くなる代わりに回路規模が膨大になる．順序回路により処理の一部を逐次処理すれば，実行時間は長くなるが回路規模を縮小することができる．

Refinement Procedure は M のいずれかの行の全要素が '0' になると FAIL exit により終了する．順序回路による実装では，全ての繰返しを行う前に条件を満たせば FAIL exit して実行時間を短縮する．

(a) sub_comb を並べる個数を限定する方法

sub_comb の個数を減らすことで，回路規模を小さくする．ただし一度に修正できる m_{ij} の数が減るため実行時間は長くなる．以下に 3 つの実装方法を示す．

seq_i sub_comb を p_β 個並べ，M を行単位で修正する．M の行数 (p_α) だけ逐次処理を行う．

seq_i_j sub_comb を 1 個だけ用い，M を要素単位で修正する．M の要素数 $(p_\alpha p_\beta)$ だけ逐次処理を行う．

seq_j sub_comb を p_α 個並べ，M を列単位で修正する．M の列数 (p_β) だけ逐次処理を行う．

(b) sub_comb の性質を利用する方法

式 (2) の $(\forall x)$ にあたる n 入力 AND に注目する．x による逐次処理を行い，回路規模を削減する．n 入力 AND は n 個の入力のうち 1 つでも '0' が含まれると他の入力に関係なく出力が '0' になるので，出力が '0' と決まった時点で繰返しを止める．以下に 3 つの実装方法を示す．

seq_x x による逐次処理を行い，M を全要素同時に修正する．R の行数 (p_α) だけ逐次処理を行う．

seq_i_x x による逐次処理を行い，M を行単位で修正する．M の行数と R の行数の積 (p_α^2) だけ逐次処理を行う．

seq_j_x x による逐次処理を行い，M を列単位で修正する．M の列数と R の行数の積 $(p_\alpha p_\beta)$ だけ逐次処理を行う．

4　評価

各実装方法について $(p_\alpha,\ p_\beta)$=(15, 15) まで扱える回路を設計し，回路を Lucent 社の OR2C シリーズ FPGA にテクノロジ・マッピングして，回路規模 (PFU 数) と動作周波数を見積る．

次に C 言語で実装したハードウェア・シミュレータから，部分グラフ同型判定の実行クロック数を求める．このクロック数と前述の動作周波数から，同型判定の実行時間を見積ることができる．実行時間は入力グラフに依存するが，ここでは辺密度 ed_α, ed_β を (ed_α, ed_β)=(0.4, 0.4) とした場合について，連結なグラフをランダムに 100 パターン生成し実行時間の平均値を求めた．

実装方法の評価には AT 積を用いる．AT 積は回路面積 (Area) と実行時間 (Time) の積で，AT 積が小さいほど優れた回路であると言える．表 1に各実装方法の PFU 数，動作周波数，実行時間，AT 積を示す．図 2には，実行時間と回路面積の関係を示す．

表 1: 各実装方法の評価結果

実装方法	PFU	Freq.[MHz]	Time [sec]	AT 積
comb	2754	22.47	3.89×10^{-2}	1.07×10^{2}
seq_i	1770	27.62	1.04×10^{-1}	1.84×10^{2}
seq_i_j	467	34.22	8.88×10^{-1}	4.15×10^{2}
seq_j	754	28.38	1.37×10^{-1}	1.04×10^{2}
seq_x	671	23.08	1.16×10^{-1}	7.78×10^{1}
seq_i_x	529	33.98	6.31×10^{-1}	3.34×10^{2}
seq_j_x	555	33.98	9.37×10^{-1}	5.20×10^{2}

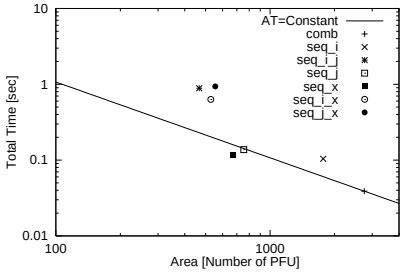

図 2: 各実装方法の実行時間，面積

図 2の直線は組合せ回路 (comb) の AT 積を基準とした AT積=一定の直線である．この直線より下に位置する seq_j, seq_x は，逐次化によって comb より AT 積が良くなっている．

5　おわりに

Ullmann の提案 [1] に従い組合せ回路で実装すると回路規模が過大になり実装は困難である．一方，順序回路による実装では Ullmann のアルゴリズムの性質を利用して，実用的な回路規模で Ullmann の提案よりも AT 積の良い実装を得ることができる．

参考文献

[1] J. R. Ullmann, "An Algorithm for Subgraph Isomorphism," *J. ACM*, Vol. 23, No. 1, pp. 31–42 (1976).