# Field Programmable Gate Array を用いた探索問題の高速化

1H－2

高木正昭 (筑波大学) 丸山勉 (同)　星野力 (同)

## 1. はじめに

探索問題の計算は，データ量に応じて，膨大な計算時間を必要とする．しかし，多くの探索問題には，並列性が内在しており，ハードウェア化することで，著しい高速化が期待できる．

再構成可能なLSIチップであるField Programmable Gate Array (FPGA)は，近年，高集積化及び高速化が進んでいる．それに伴い，マイクロプロセッサにFPGAを付加したシステムが考えられ，FPGAのハードウェアアクセラレータとして応用が模索されてきている．FPGAを用いることで，マイクロプロセッサの苦手な点を補い，データに適した高速化が可能となる[1)]．

今回，典型的な木探索問題のプログラムをハードウェア化する．その際，並列化，パイプライン化を施し，高速化を図り，ソフトウェアとの速度比較を行う．

## 2.Field Programmable Gate Array(FPGA)

FPGAとは，プログラムすることによって，論理回路が再構成できるLSIチップのことである．ハードウェア記述言語でプログラムを記述し，それをコンパイルし，FPGAにダウンロードすることによって，論理回路をLSI上に構成することができる．今回，ALTERA社のFLEX10Kシリーズを用いた[2)]．

## 3. 探索問題

木探索問題を解くアルゴリズムをナップサック問題に適用し，ハードウェア化を行った．

ナップサック問題とは，“価値”と“重さ”という2つの価値を持ったオブジェクトの集合から重量制限内で，価値の和が最大になるようにオブジェクトを選択するという整数計画問題である．このナップサック問題の探索空間をオブジェクトを選択するか/しないかの2分木(図1)にすることによって，木探索問題として，解くことができる．

図2は，今回，ハードウェア化したプログラムである．このプログラムは，縦型探索を行っていき，今までの最高価値と比較して，改善が見込めないときは，探索をやめ枝刈りを行う．プログラムの構造は，tail recursionになっており，2個所で再帰が行われている．最初の再帰部分で，縦型探索のための子節点を探索していき，終端までたどり着いたとき，または，枝刈りが起きたとき，もう一つの再帰部分が実行される．

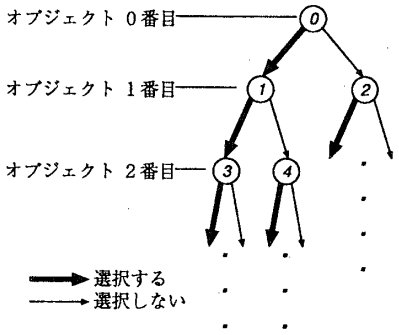


図 1: 2分木で表された探索空間

```
try(int i, int tw, int av, int obj_set)
{
/*obj[i].v,obj[i].wは、i番目の　*/
/*オブジェクトの価値と重さ、maxv, */
/*max_obj_set,obj[i]はグローバル変数*/

 int av1, j;
 av1 = av-obj[i].v;

 if(tw+obj[i].w<=limw){
  if(i<N-1)
   try(i+1,tw+obj[i].w,av,obj_set|(1<<i));
  else{
   if(av>maxv){
    maxv=av;
    max_obj_set=obj_set|(1<<i);
   }
  }
 }
 if(av1>maxv){
  if(i<N-1){
   try(i+1,tw,av1,obj_set);
  }else{
   maxv=av1;
   max_obj_set=obj_set;
  }
 }
}
```

図 2: ハードウェア化したCプログラムの断片

## 4. ハードウェア化

図2の処理は，同時に実行しうる全ての演算を同時に行うことで，メモリアクセス以外の処理を2クロックで，

---

High speed computation of a search problem with FPGA

Masaaki Takagi, Tsutomu Maruyama and Tsutomu Hoshino　University of Tsukuba

実行することができる．

メモリアクセスとしては，データの読み出しとスタックへの読み出し / 書き込みに計 3 クロック必要となる．従って，図3のような5段のパイプラインとなる．

stage2 において，図 2中の 2 つの再帰呼出しに対する条件判断が行われる．両方の再帰呼出しに対する条件が成立した場合は，stage4 において，2 つ目の再帰呼びだしに対する環境がスタックに待避され，一つ目の環境に対する計算が stage0 から，実行される．どちらか一方のみが，実行される場合には，スタックへのアクセスは行われず，その環境が stage0 へ送られる．両方の条件が，満たされなかった場合は，stage4 において，スタックに格納されていた環境が読み出され，stage0 から処理される．スタックが，空の時は，処理が終了する．

このような処理を行った場合，同時に動作するのは，stage0 ～ stage4 のうちの 1 つだけである．そこで，スタックに待避されている環境に対する計算を図 4に示すように，次々と pop することによって，最大，stage 数のパイプライン処理を実現することができる．これは，図 1において，探索木の異なる節点を同時に計算することに相当する．

同じパイプラインをいくつか作ることにより並列化ができるが，今回は，演算レベルでの並列化だけを行った．

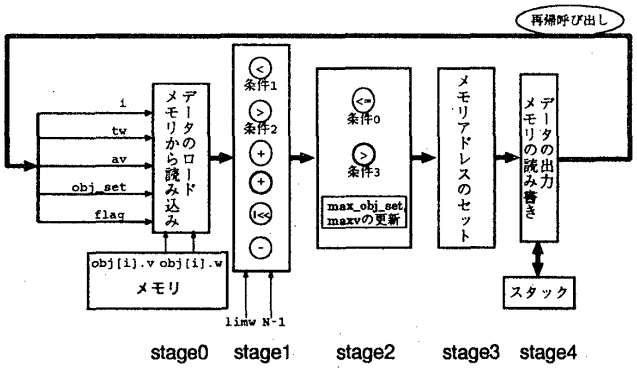


図 3: パイプライン

## 5. 結果

i:4bit, tw, av:16bit, obj_set:32bit として，ハードウェア化した結果，動作周波数は，20MHz となった．

オブジェクト数を 30，重さと価値をランダムに設定

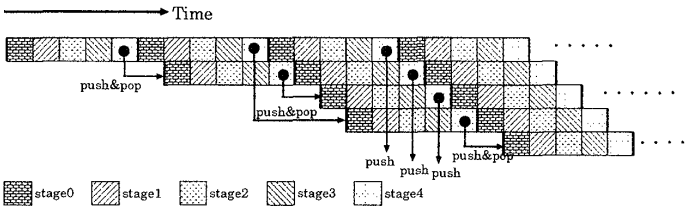


図 4: 各ステージの動き

し，ワークステーション (SUN Ultra Sparc 200MHz) で実行した場合と FPGA で実行した場合の性能を比較した．その結果，表 1のようになった．

表 1: 実行時間 (秒) と速度向上率

| | 処理時間 | 速度向上率 |
|---|---|---|
| WS | 0.302sec | 1 |
| FPGA | 0.039sec | 7.7 |

パイプライン化とそれによる並列化，および命令レベルでの並列化により，7.7 倍という速度向上が得られたと考えられる．

## 6. おわりに

今回，探索問題を解くアルゴリズムのためのパイプライン化手法を示しつつ，ハードウェア化を行った．その結果，ワークステーションと比較して，約 7.7 倍の速度向上を得ることができた．

FPGA をハードウェアアクセラレータとして用いるときの問題点として，FPGA の設計の困難さと動作周波数の遅さが上げられる．そのため，設計者の熟練と慎重な最適化が必要になってくる．

今後，この問題を解決するため，ハードウェア化するために今回行った一連のデザインフローを自動化していく予定である．

## 参考文献

1) Timothy J. Callahan and John Wawrzynek: Instruction-Level Parallelism for Reconfigurable Computing,FCCM 1998

2) ALTERA:FLEX 10K Embedded Programmable Logic Family Data Sheet,June 1996