パルテノン研究会 ASIC デザインコンテスト 規定課題

1 概要

CPU とコンパイラはディジタルシステムの基本技術であり、これらの基本的な動作を理解 することはディジタルシステム設計者としてのスタートアップとして必須といっても過言ではない。パルテノン研究会 ASIC デザインコンテストでは、エンジニアを目指す学生諸君が容易に取り組める初級コンテスト課題として 16 ビット CPU を提示する。本課題は、アーキテクチャ上の工夫、ドキュメント、プログラム実行時の消費エネルギー、設計の信頼性などを審査対象物とする。この課題は初級コンテストであるため、対象を大学院修士（博士前期）課程以下の学生に限定する。この課題で要求される技術は下記のものとなる。

1. ソースコードで定義される課題プログラムを実行する CPU の開発
2. 同プログラムをコンパイル可能なコンパイラ開発
3. 論理シミュレーションによる動作検証
4. エネルギーを最小化するハードウェア・コンパイラ改良

プログラムの実行サイクル数と実行結果の検証は論理シミュレーションによって行い、エネルギー・実行時間は、PARTHENON（ライブラリは DEMO 社の demo ライブラリ）による論理合成結果の消費電力・最大遅延時間を用いて算出する。実行サイクル数はシミュレーションスタート時点から停止命令実行までのクロック数でカウントする。[1]コンテストチャレンジャは、審査用設計ドキュメントの他に、CPU の論理記述ファイル、コンパイラ・アセンブラのソースファイル、Makefile 等評価に必要なデータ一式を電子ファイルにて提出する。シミュレーション・論理合成に必要な環境・手順を審査用ドキュメント中に明記すること。（審査において再現性がないと判断したときは、採点ポイントを大きく減点する）以下の点に留意すること。

- ドキュメント審査と動作検証は独立して行われるので、ドキュメントには、評価可能なだけの、十分な情報量を持たせること。
- 課題ソースコードをコンパイルし、論理シミュレーションを実行するまでの一連の流れをコマンドラインから make コマンド等で一括実行可能なこと。
- 審査用非公開課題プログラム (foo.sc) のシミュレーション実施方法を明示すること。
- 論理シミュレーションにおいて、適切なデータ表示がなされてプログラムの実行結果を検証できること。

2 課題プログラム

二つのプログラム（バブルソート・クイックソート）をコンテスト課題プログラムとする。課題プログラムのソースコードは変更してはならない。コンテストは二つの課題プログラムと、審

[1]論理シミュレーションツールは、フリーソフトウエアもしくは非商用ライセンスなど評価者が無償利用可能なものに限定する。

査委員が用意する非公開のプログラム計 3 本のプログラムの消費エネルギーの相乗平均値で評価し、この値が小さいほど優秀な設計とする。評価は、この値のほか、アーキテクチャの工夫、ドキュメンテーション、信頼性などを複数の審査員の採点による点数を元に行う。コンテスト参加者は提示されたソースコードをコンパイル可能なコンパイラを作成し、コンパイラが出力する機械語を論理シミュレーションに用いる。初級課題であることを考慮し、コンパイラ、アセンブラを含む例題パッケージを用意した。例題パッケージは CPU の論理ファイルとシミュレーション環境、およびアセンブラ、コンパイラのソースコード群から構成され、IP-ARCH 社のホームページ (http://wwww.ip-arch.jp) からダウンロード可能とする。例題のプロセッサアーキテクチャおよびコンパイラの詳細は同ページの参考文献を参照のこと。例題コンパイラ・アセンブラは yacc と lex で文法を記述し、C でコード生成を記述したもので、ターゲット言語は C 言語の簡単なサブセットである。コンパイラの言語仕様を次にまとめる。

CPU アーキテクチャ
- 命令・データのアドレス空間を分離したハーバードアーキテクチャー
- 命令・データとも 1 ワード (16 ビット)1 アドレスのワードマシン
- 命令メモリ・データメモリのアクセスには 1 クロックサイクルを要する

変数 大域変数
- メモリ上の 1 番地から順に格納される `a`〜`z`

  変数には添字を付けることができるが、コンパイラは、配列範囲の場所を確保しない。`a[0]` は `a` と等しく、`a[26]` は `z` と等しい。
- 任意の識別子（ソースに出現順に z の後ろに配置される）。配列の大きさを示すことで、配列として利用可能。`int AbcDef[10];` のような配列宣言により、配列範囲のメモリ領域を確保する。
- mem 配列

引数 関数の宣言時に括弧内に宣言した識別子は仮引数となる

局所変数 関数の先頭で宣言した int 型識別子は、その関数の局所変数となる。

関数 関数は整数型か void 型任意の関数名が定義できる。

演算 `+ - < > >= <= == & |~ >> <<` が利用可能。

構文 `while if else for return break halt` が利用可能。

halt 実行時にはプロセッサを停止させ、シミュレーションをストップすること。

2.1 課題１：クイックソート

図 1 に課題 1 のクイックソートプログラムを示す。このプログラムは、クイックソートを行う関数、項目のスワップを行う関数を有し降順に z 変数上に配列を作成した後、その値を昇順に並び替える。

2.2 課題２：バブルソート

図 1 に課題 2 のバブルソートプログラムを示す。このプログラムは、降順に z 変数上に配列を作成した後、その値を昇順に並び替える。

```
void qsort(left, right) {
 int ileft,jright,pivot;
 ileft=left;
 jright=right;
 pivot=z[(left+right)>>1];
 while(1) {
   while(z[ileft]<pivot) ileft++;
   while(pivot < z[jright]) jright--;
   if(ileft>=jright) break;
   swap(ileft,jright);
   ileft++;
   jright--;
 }
 if(left<ileft-1) qsort(left,ileft-1);
 if(jright+1<right) qsort(jright+1,right);
 return;
}

void swap(ain, bin) {
 int temp;
 temp=z[ain];
 z[ain]=z[bin];
 z[bin]=temp;
 return;
}

for(i=0;i<20;i++) z[i]=20-i;
qsort(0,19);
halt;
```

図 1: クイックソート課題のソースコード

```
n=20;

for(i=0; i<n; i++) z[i]=n-i;

for(i=0; i<n-1; i++)
  for(j=n-1; j>i; j--)
    if(z[j]<z[j-1]) {
        w=z[j];
        z[j]=z[j-1];
        z[j-1]=w;
     }

halt;
```

図 2: バブルソート課題のソースコード

表 1: 例題 CPU による課題の実行クロック数

課題	クロック数
qsort	14928
sort	43681
foo	32100(not measured yet)

表 2: 例題 CPU 論理合成結果

class_name	snx
power	5783.7
area	809.04
gates	3006

2.3 課題３：ブラインド課題

本課題のプログラムは開示されない。コンパイラのすべての機能を正しく実装すれば、ブラインド課題の実行には支障はない。

3 コンテスト審査数値の計算手順

1. 合成した結果の最大遅延時間でプロセッサを動作させた時の各プログラムの実行時間を求める。
2. 最高クロック周波数で動作する時の各消費電力と実行時間から各プログラムの実行に必要なエネルギーを求める。
3. 実行エネルギーの相乗平均を求める。

各プログラムの実行クロック数、合成結果が表 1,2 のようになっていたとする。(ここで foo のクロック数は仮の値である）最大立上がり遅延時間が 74.0nS とすると実行時間・エネルギーの計算は次のようになる。クロック周期は 74.0nS であるため、実行時間 の相乗平均値は

$$T = \sqrt[3]{14928 \times 43681 \times 32100} \times 74.0 \times 10^{-9} = 2.0 \times 10^{-3}[S]$$

消費電力 P は 5783.7μW/MHz なので、実行エネルギーの相乗平均値は、

$$T \times P = \frac{2.0 \times 10^{-3} \times 5783.7 \times 10^{-6}}{74.0 \times 10^{-9} \times 10^{6}} = 1.6^{-4}[J]$$

となる