4L－8　専用プロセッサ設計支援システム(SYARDS)における ハードウェア指向・ソフトウェア指向合成法の評価

樋渡 仁　　白井 克彦

早稲田大学 理工学部

1　はじめに

近年、高位合成の分野では、動作記述をハードウェアで実現する部分とソフトウェアで実現する部分に分割し、最終的にこれらを合成する協調設計 (codesign) という設計法が注目を集めている。われわれは、以前からアルゴリズムによる動作記述を入力とした専用プロセッサ設計支援システム (SYARDS) の提案を行なってきた。ここでは、これらの状況を考慮しながら、SYARDS における協調設計について、適当な例を用いて評価を行なう。

2　協調設計

従来の高位合成は、入力されたアリゴリズム記述をできるだけ高速に実行するように設計することを目的としてきた。しかし、汎用プロセッサの高速化にともなって、アルゴリズム記述のある部分をソフトウェア (マイクロプロセッサ) で実行し、残りの部分をハードウェア (ASIC) で実行することができれば、より効率的な設計が可能になるという考え方が出てきた。これが、ハードウェア/ソフトウェア協調設計 (hardware/software codesign) と呼ばれる設計方法である。

協調設計における最も重要な概念は、すべてソフトウェアで実現した設計とすべてハードウェアで実現した設計の間で、性能やコストに関する制約条件を満たすように設計を行なえるということである。すなわち、比較的緩い制約条件であれば、大部分をソフトウェアで実行し、比較的厳しい制約条件であれば、大部分をハードウェアで実行するといった選択を設計者自身が行なうことができる。

3　SYARDS における協調設計

一方、設計支援システム SYARDS においても、次のような設計手法が提案されている。

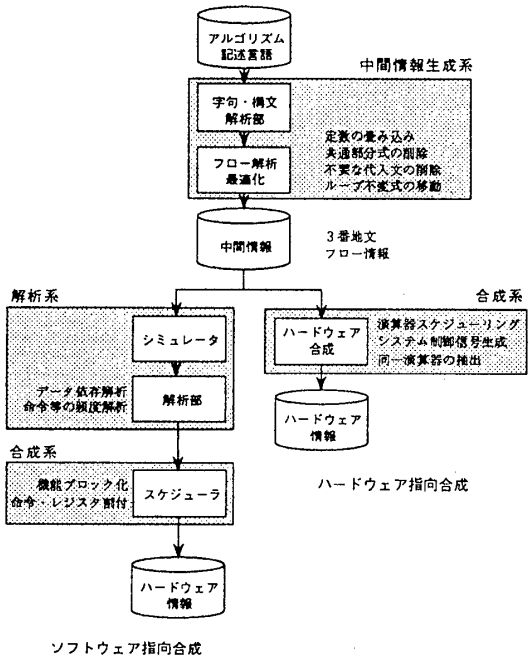

図 1: システム構成図

○ ハードウェア指向合成法
○ ソフトウェア指向合成法
○ ハードウェア/ソフトウェア協調合成法

また、各設計手法は、図1に示すような設計フローとして統合されている。

各設計法について簡単に説明する。ハードウェア指向合成法は、アルゴリズム記述から得られた中間情報を ASAP スケージュリングし、各コントロールステップでどのような命令が実行されるかを決定する。さらに、実行に必要なハードウェア要素を抽出し、状態遷移にもとづいたハードウェア記述言語を用いて結果を出力する。

ソフトウェア指向合成法は、一般にテンプレート法と呼ばれる方法で、あるテンプレートのプロセッサを利用してアルゴリズムの実行に必要なハードウェア要素のみを合成する。さらに、ハードウェア/ソフトウェア協調合成法は、命令列の中から高頻度に出現する同

Evaluation of Hardware-oriented and Software-oriented Approach in Design System for Special Purpose Processors (SYARDS)

Jin HIWATASHI and Katsuhiko SHIRAI
Department of Electrical Engineering, Waseda University

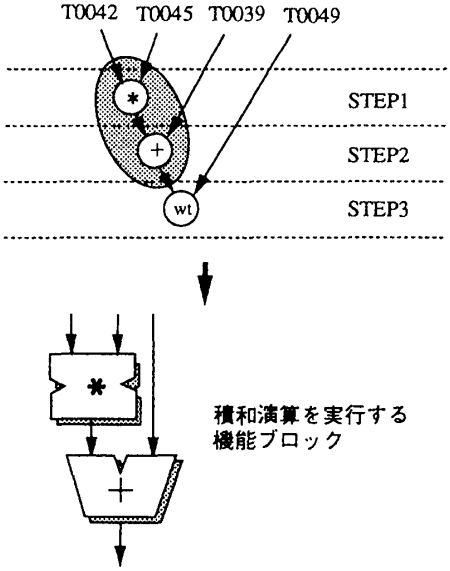

図 2: ハードウェア/ソフトウェア協調合成法

じ動作を行なう命令群を抽出し、機能ブロック (複合命令) として 1 命令に合成することによって、より高機能なプロセッサを設計する (図 2 参照)。

4 評価

設計支援システム SYARDS における協調設計の可能性を探る目的で、既存の各合成法を用いて同じ問題を設計し比較することを考える。ここでは、画像処理の代表的なアルゴリズムである離散コサイン変換 (DCT) を取り上げて、各合成法においてどのような設計が行なわれたかを比較する。

また、DCT のアルゴリズム記述は、システムへの入力言語である SYARDS C で記述されている (図 3 参照)。

```
#include <stdio.h>
#include "ioport.h"
.....
main(int argc, char **argv)
{
   int  g[64], f[64];
   short fcut[64], gcut[64];
   short image[64];
   short i, x, y, u, v;
   .....
   /* 1st IDCT */
   for (x = 0; x < 8; x++) {
      for (v = 0; v < 8; v++) {
         for (u = 0; u < 8; u++) {
            g[x*8+v] += image[v*8+u] * idct[u*8+x];
            .....
```

図 3: アルゴリズム記述 (DCT)

DCT を実現するために、各合成法を用いた設計が必要とした記憶要素を表 1 に示す。ただし、ここで 1word とは 32bit を表す。表 1 において、ゲート数の中には、RAM と ROM の情報は含まれていない。

表 1: 必要な記憶要素の比較

合成法	ゲート数 [gate]	RAM [Kword]	ROM [Kword]	レジスタ [word]
HW	8834	3	0	10.75
SW	17241	0.5	0.25	17
HW/SW	16541	0.5	0.25	15

次に、各設計が与えられたアルゴリズムを実行するのに必要な実行時間を表 2 に示す。やはり、DCT というアルゴリズム自体がハードウェア指向合成法に適していたため、ソフトウェア合成法との間に大きな性能の差ができてしまった。

表 2: 実行時間の比較

合成法	ステップ数 [step]	クロック周期 [ns]	実行時間 [μ s]
HW	3956	25.910	102.49
SW	33966	75.440	2562.39
HW/SW	30192	78.551	2371.61

5 おわりに

設計支援システム SYARDS を用いて、異なる合成法で同じハードウェアを設計し、その性能やコストに関する評価を行なった。この結果から、SYARDS において協調合成を行なうには、さらなる合成法の確立が必要なことが分かった。

参考文献

[1] 北畠 宏信, 白井 克彦: "高位仕様記述からの専用プロセッサ設計における機能設計について," 情報処理学会研究報告第 60 回設計自動化研究会, pp.25–32 (1991).

[2] 池永 剛, 白井 克彦: "高級言語により記述されたアルゴリズムを実現する専用プロセッサ設計支援システム," 情報処理学会論文誌, vol.32, pp.1445–1456 (1991).