入力プログラムと中間表現の実行に基づく高位合成システムのテスト

Testing of high level synthesizer based on execution of input programs and their intermediate representations

石守 祥之 †　石浦 菜岐佐 †
Yoshiyuki Ishimori　Nagisa Ishiura
西村 啓成††　神原 弘之‡　冨山 宏之§
Masanari Nishimura　Hiroyuki Kanbara　Hiroyuki Tomiyama

1　はじめに

高位合成は, プログラミング言語等の動作記述からハードウェア (以下 HW) を合成することにより, LSI の設計の効率化を図る技術であるが, 高位合成システムそのものの信頼性の確保も一つの課題となる. 高位合成システムの検証手法としては, 合成における各変換の正当性を形式的手法によって示す方法 [1] が提案されているが, 合成系全体への適用は未だ難しいと考えられる. このため, 合成した HW のシミュレーションに基づく方法が一つの現実的な手法と言えるが, 何を HW の動作の期待値とし, 照合をいかに行うかが課題となる.

本研究では, 高位合成の各過程で生成される中間表現を C プログラムに変換し, その実行結果を合成の入力として与えられた C プログラムと比較するテスト手法を提案する. 本研究のテストはマルチスレッドで実行するが, スレッド間の同期の実装に, busy loop を用いる手法と条件変数を用いる手法を示す.

2　高位合成システム CCAP

我々は, ANSI-C プログラム中の指定した関数を, ソフトウェア (以下 SW) として CPU で実行される他の関数から呼び出せる形の HW に合成する高位合成システム CCAP (C Compatible Architecture prototyper) [2] の開発を行っている. 図 1 に CCAP が合成するシステムの構成を示す. どの関数を HW で実行し, どの関数を SW で実行するかはプラグマにより指定する. SW と合成された HW は記憶空間を共有し, SW 及び HW 間のデータ授受及び HW の呼び出しは, 主記憶上に配置されるグローバル変数へのアクセスにより実現する.

CCAP の合成処理の流れを図 2 に示す. CCAP は合成の過程で, CDFG (C プログラムを抽象構文木により表現し, 基本ブロックをデータフローグラフで表現する), STL (計算を HW モジュールによる関数計算, レジスタへの転送, 及びこれらを制御する状態遷移により表現する) 及び RTL (STL に演算器の信号線等の詳細の情報を追加したもの) と呼ぶ中間表現を生成する. CDFG には

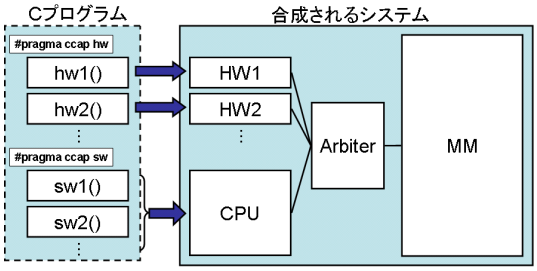


図 1: CCAP が合成するシステムの構成

さらに, cdfg (プログラムをそのまま変換したもの), low (cdfg 中の if や loop などの構文をすべて条件付の goto 文に変換したもの), dfa (データフロー解析を行ったもの), sch (演算のスケジューリングを行ったもの), bnd (演算器やレジスタのバインディングを行ったもの) 等のサブクラスが存在する.

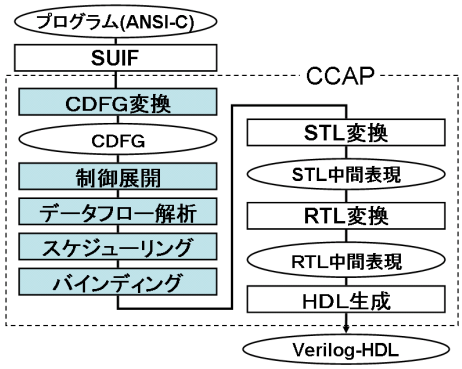


図 2: CCAP の処理の流れ

3　高位合成システムのテスト手法

本稿では, CCAP の各中間表現の動作を元の C プログラムと比較するテスト手法を提案する. テストの流れを図 3 に示す. HW に合成する部分を hw.c とし, SW として実行する部分を sw.c とする. hw.c と sw.c をコンパイルし, 実行して得られる結果を期待値とする. 次に hw.c を CCAP に入力し, 各変換処理で得られる中間表現 (cdfg 等) を, その動作をシミュレーションする C プログラム (cdfg.c 等) に変換する. これをコンパイルして得られるシミュレータ (sim_cdfg 等) の実行結果を期

†関西学院大学 大学院理工学研究科
††同上 (現 株式会社ルネサスソリューションズ)
‡京都高度技術研究所
§名古屋大学 大学院情報科学研究科

待値と比較する.

テストプログラムの簡単な例を図 4 に示す. SW として実行する main で HW 化する関数 hw_mult が正しく乗算を行うか, hw_accum が正しくグローバル配列にアクセスするかをテストしている.

本手法では, この出力の比較によりテストを行うので, 期待値の準備は不要になる.

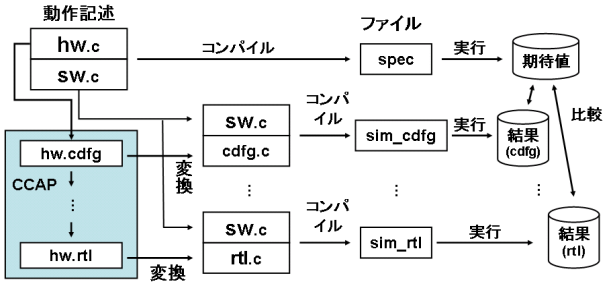


図 3: テストの流れ

4 SW/HW シミュレーションのマルチスレッド実装

本テスト手法では, CCAP が合成するシステムのシミュレーションをマルチスレッドで実装する. 即ち, SW に 1 スレッドを, HW 化される関数毎に 1 スレッドを割り当て, システムの動作をシミュレーションする.

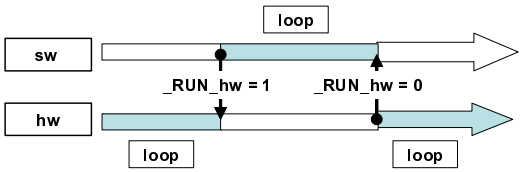
CCAP では, 関数間の待ち合わせを busy loop で実装するため, それをそのまま 3 節の方法で変換するとシミュレータ上でも busy loop による待ち合わせとなる. Busy loop はスレッドが待機状態時にも CPU を消費して非効率的となる可能性があり, シミュレータ生成時に条件変数を用いる待ち合わせ (条件変数が送信されるまで待機中のスレッドはサスペンドされる) への変換も選択できるようにする.

5 実験

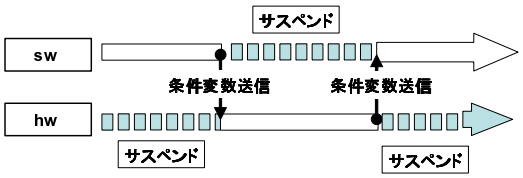
スレッドの待ち合わせを, busy loop による手法と条件変数による手法を用いる場合で, 実行時間にどの程度

```
#define N 64
int a[N];
#pragma ccap hw
int hw_mult(int a, int b) {return a*b;}
void hw_accum(){
    for( i = 1; i ≦ N ; i++) a[i] += a[i-1];
}
#pragma ccap sw
int main(void) {
    //乗算の合成のテスト
    for( i = -256; i < 256; i +=7 ){
        for( j = -256; j < 256; j +=7 )
            printf("%d\n", hw_mult(i, j));
        }
    }
    //グローバル配列のアクセスのテスト
    for( i = 0; i < N ; i++ ) a[i]=i;
    hw_accum();
    for( i = 0; i < N ; i++ ) printf("%d\n", a[i]);
    return 0;
}
```

図 4: テストプログラムの例



(a) busy loop による手法



(b) 条件変数 による手法

図 5: スレッド間の待ち合わせ

の差異が現われるかを測定する実験を行なった*. 結果を表 1 に示す. 実験はシングルコア環境 (CPU: Intel Pentium M 1.20GHz, メモリ: 0.25GB) 及びマルチコア環境 (CPU: Intel Core2 T5500 1.66GHz, メモリ: 0.99GB) 下で行った. SW+HW はスレッド数を call depth は HW の呼出しによる深さを表す. シングルコア環境下においては, スレッドの条件変数を用いるほうが高速に実行できる. マルチコア環境下では条件により異なるが条件変数を用いる方が安定的に高速に実行することができる.

表 1:実行時間 (sec)

コア数	SW+HW	call depth	busy loop	条件変数	比
1	1+1	1	16.2	0.180	90.0
	1+8	1	3.368	0.550	6.124
	1+8	8	4.104	0.270	15.189
2	1+1	1	3.301	0.440	7.502
	1+8	1	1.030	0.855	1.205
	1+8	8	0.999	0.855	1.211

6 むすび

本稿では, 入力プログラムと中間表現の実行に基づく高位合成システムのテスト手法を提案した. CCAP 用のテストスイートの作成, Testgen [3] を用いたテスト等が今後の課題として挙げられる.

謝辞 本研究に際し, テストの作成に御協力頂いた奥野裕氏及び関西学院大学石浦研究室の諸氏に感謝致します.

参考文献

[1] P. Ashar, et al.: “Verification of RTL generated from scheduled behavior in a high-level synthesis flow,” in *Proc. ICCAD ’98*, pp. 517–524 (Nov. 1998).

[2] M. Nishimura, et al.: “High-level synthesis of variable accesses and function calls in software compatible hardware synthesizer CCAP,” in *Proc. SASIMI 2006*, pp. 29–34 (Apr. 2006).

[3] http://ist.ksc.kwansei.ac.jp/ ishiura/testgen/

*[実行環境] OS: Microsoft Windows XP Professional SP2, コンパイラ: gcc-3.4.4 (Cygwin)