A-03

MIPS アセンブリを中間表現とする高位合成
High-Level Synthesis Using MIPS Assembly Programs as Intermediate Representation

池上 達也 † Tatsuya Ikegami
石浦 菜岐佐 † Nagisa Ishiura

1 はじめに

高位合成 [1] は, C 言語等で書かれた動作記述からレジスタ転送レベルの回路記述を生成するハードウェアの設計技術であり, 大規模化・複雑化が進む LSI の設計効率を向上させる技術として実用化が進められている.

我々の研究グループでは, C プログラムからソフトウェアとハードウェアより成るシステムを効率的に合成すること, 及び入力記述をできる限り ANSI-C に近づけることを目的とした高位合成システム CCAP [2] の開発を進めている. CCAP は入力として与えられた C プログラム中の指定した関数を, ソフトウェアとして CPU で実行される他の関数から呼び出し可能なハードウェアに合成することができる. ポインタを用いたメモリアクセスをハードウェアに合成可能であり, これによってグローバル変数を介したハードウェア・ソフトウェア間のデータオブジェクトの受け渡しができる. しかし, 現時点では ANSI-C に準拠したすべてのプログラムを合成できるわけではなく, 構造体や可変配列等, 合成が困難な記述が存在している. 高位合成システムでより広範なプログラムを扱うための手法として, コンパイラが生成するアセンブリコードを入力とした高位合成手法 [3, 4] が提案されている. しかし従来手法はいずれも, 単体のハードウェアの合成を対象とするものであり, ハードウェアに合成される関数間での呼び出しを考慮したものではない.

そこで本研究では, より広範な C プログラムが合成可能で, かつ合成されるハードウェア間での関数呼び出しが可能な高位合成の手法を提案する. 本手法により, 従来よりもさらに広い範囲の C プログラムを合成の対象とすることが可能になる.

2 高位合成システム CCAP

CCAP は, 入力として与えられた C プログラムの一部の関数を, 他の関数から呼び出し可能なハードウェアに合成する高位合成システムである. CCAP は, 入力となる C プログラムから中間表現として CDFG (Control Data Flow Graph) を生成し, そこからスケジューリング, バインディング等の処理を行って Verilog HDL を生成する.

ハードウェア化された関数の呼び出しは, グローバル変数へのアクセスを介したポーリングにより実現する. 各関数に, 引数用, 起動用, 返り値用のグローバル変数を作成する. 呼び出し側では, まず引数を引数用グローバル変数に書き込み, その後起動用グローバル変数に 1 を書き込むことにより関数を起動する. 関数の実行が終了すると起動用変数の値は 0 になるので, 呼び出し側はこれを待って処理の続きを実行する. 呼び出される関数は, 起動用変数を監視し, その値が 1 になれば実行を開始する. 関数の開始時に引数用変数から引数をロードし, 終了後に返り値用変数に返り値を書き込む. 最後に起動用グローバル変数に 0 を書き込み, 制御を呼び出し元に返す.

CCAP は算術演算や, 配列, ポインタ等のデータアクセスを合成することはできるが, 構造体や可変配列, 及びローカル変数のポインタに対する演算等, 関数のフレームへのアクセスを要する構文や, switch 文等の一部の制御構造を扱うことができない. このため, リファレンスとして与えられる任意の C プログラムをそのまま合成できるには至っていない.

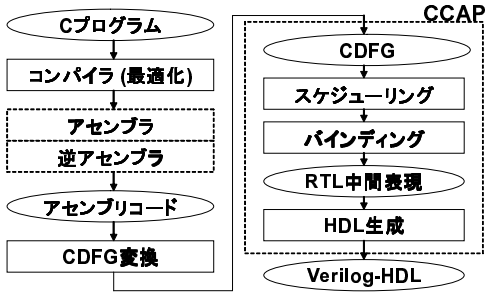
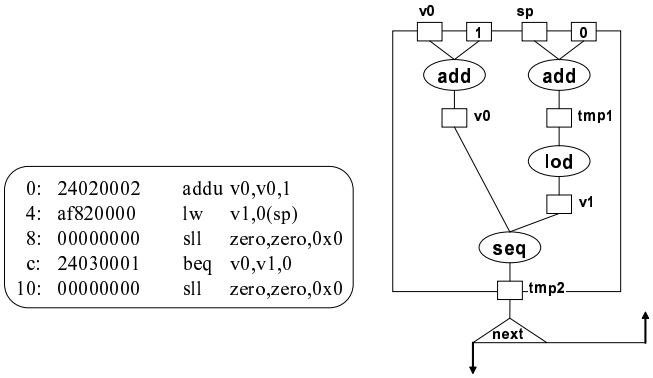


図 1 処理の流れ



(a) MIPS アセンブリ (b) CDFG

図 2 MIPS アセンブリの CDFG への変換

3 アセンブリを中間表現とする高位合成

3.1 処理の流れ

アセンブリを中間表現とする高位合成の処理の流れを図 1 に示す. C プログラムからコンパイラによって最適化されたアセンブリコードを生成し, そこから CDFG を生成して RTL 回路の合成を行う. CDFG 以降の処理は従来の高位合成手法と同様である. コンパイラが生成する疑似命令が扱い難い場合には, アセンブリコードを一旦アセンブルしてそれを逆アセンブルすることによりこれを除去する.

† 関西学院大学 理工学部 情報科学科

この方式により, C プログラムレベルの情報 (例えばループ構造) を利用できなくなるため, 合成する回路の性能は低下する可能性があるが, より広範な C プログラムを合成できることが期待できる. またコンパイラの強力な最適化機能をそのまま利用できることもメリットになる.

本研究では MIPS R3000 のアセンブリから合成を行う. MIPS は RISC であるため命令セットが小さく, 各命令の高位合成の演算への変換が容易である. また, コンパイラには gcc を用いる.

3.2 MIPS アセンブリの CDFG への変換

入力となるアセンブリコードは基本ブロックに分割し, 各基本ブロックを DFG の形に変換する. 図 2 に MIPS アセンブリから CDFG への変換例を示す. 基本的には, MIPS の 1 命令を対応する CCAP の CDFG の 1 演算節点に置き換える. 例えば, 図 2(a) の 0 行目の addu (符号無し整数の加算) は, 図 2(b) 左上の ADD 演算節点に変換する. 命令によっては MIPS の 1 命令を CCAP の複数演算に変換することもある. 図 2(a) の 4 行目の lw 命令はメモリーアクセスに先立ち実行アドレスの計算を行うので, 図 2(b) 右上のように SP とオフセットの加算と, その結果をアドレスとするメモリー読み出しの 2 演算に変換する. 図 2(a) の 8 行目の sll 演算は nop を表しているため削除する. 図 2(a) の c 行目のような分岐命令は, 分岐の有無を表す値と, 分岐先の DFG を表すポインタに変換する. 分岐命令の delay slot にある命令 (ここでは nop なので削除する) は, 分岐命令より上の基本ブロックに含める.

レジスタへのアクセスは, 一旦抽象的な「値」へのアクセスに変換し, バインディング段階で, 関数ローカルなレジスタに再割り付けする. このため, 合成後のレジスタ数が MIPS のレジスタより多く (少なく) なることがある. また, 静的単一代入 (SSA) 変換と等価な処理を行っているため, アセンブリでは同じレジスタに格納されている値が合成後には別のレジスタに格納されることもある.

本手法では, 入力となるアセンブリは C プログラムから gcc によって生成されると仮定している. これに伴い, レジスタの使用法に制限を設けている. レジスタ at, k0〜k1 は, コンパイラでは使用しないレジスタなので, 本手法では処理の対象としない. gp はグローバル変数のアクセスに使われるが, CCAP ではグローバル変数のアドレスは HDL 生成時に決定するので, objdump から得られるシンボルだけを記録し, gp は CDFG から削除する. この他, 4 章で述べるように, 関数呼び出しに関連して, v0, v1, a0〜a3, sp, fp, ra の使用法にも制限を設ける.

本手法では, MIPS R3000 の全命令 74 命令のうち, 52 命令が合成できる. 合成できない命令は, 特殊命令 (break, syscall 等), レジスタを用いたジャンプ命令 (jr, jalr 等), Branch-and-Link 命令 (bltzal, bgezal 等), コプロセッサ命令である.

4 関数呼び出しの合成とレジスタの扱い

関数呼び出しの方式は 2 節で紹介した CCAP と同様であり, グローバル変数を用いることによって実現する. すなわち, 引数や返り値等の値を格納したレジスタの値をグローバル変数を介して受け渡し, 起動用グローバル変数によるポーリングにより制御を受け渡す. この際, アセンブリコードがコンパイラの関数呼び出し規約に従っていることを前提とすることにより, 値を受け渡すべきレジスタを限定する. 関数呼び出しに必要なデータの受け渡しをレジスタ毎にまとめる.

- a0〜a3
  引数渡しに使われるレジスタ a0〜a3 の値は, 関数起動の直前に引数用グローバル変数に書き込む. 被呼び出し側では, 関数の開始時に, ここから値をレジスタにロードする. 4 個を超える引数やレジスタに格納できない引数に対しては, 関数フレームで値を受け渡すコードが生成されているので, これをそのまま合成する.
- v0, v1
  関数の返り値を保持するレジスタ v0, v1 の値は, 返り値用グローバル変数を介して, 関数の実行終了時に受け渡す.
- t0〜t9
  一時変数の保持に使われるレジスタ t0〜t9 はそのままローカル変数として合成する.
- s0〜s7
  s0〜s7 は「saved registers」で, 被呼び出し側で使用するときは, レジスタの値をフレーム内に退避し, 使用後に復帰させるコードが生成される. 本手法では, レジスタは全て関数ローカルに使用するため, レジスタの退避・復帰は不要となる. 従って, s0〜s7 に関するレジスタの退避・復帰のための命令は消去する.
- sp, fp
  関数フレームのアクセスに用る sp, fp は関数呼び出し/復帰時に値を受け渡す必要がある. これについては, 関数毎ではなく全体に共通のグローバル変数を作成する. sp, fp を通常のレジスタ t0〜t9, s0〜s7 の代替として用いることはできない.
- ra
  ra は戻りアドレスを保持している. 戻りアドレスは合成されるハードウェアでは用いないので, 受け渡しは不要である. 本手法では ra はアセンブリのリターン命令の認識に利用する. 即ち"jr ra"命令を関数からのリターン命令と認識し, 関数終了後の戻り値等のストアやポーリング変数への終了値のストアの演算に変換する.

5 むすび

本稿では MIPS アセンブリを中間表現とする高位合成の手法を提案した. 現在, 本手法の実装を進めており, 算術演算, 配列, ポインタ, ハードウェア間の関数呼び出しを Verilog HDL に合成可能である. 今後は実装を進め, C プログラムから直接合成した場合とのサイクル数や回路規模の比較を行っていく予定である.

謝辞

本研究を進めるにあたり御助言・御討論を頂きました京都高度技術研究所の神原弘之氏, 名古屋大学の冨山宏之准教授はじめ, HLS プロジェクトの諸氏に感謝します.

参考文献

[1] Daniel D. Gajski, Nikil D. Dutt, Allen C-H Wu, and Steve Y-L Lin: *High-Level Synthesis: Introduction to Chip and System Design*, Kluwer Academic Publishers (1992).

[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] 尾形秀範, 北川章夫: "アセンブリレベル合成法," 信学技報, CAS2004-78 (Jan. 2005).

[4] G. Mittal, D. C. Zaretsky, and X. Tang: "Automatic Translation of Software Binaries onto FPGAs," in *Proc. 41st DAC,* pp. 389–394 (June 2004).