C_009

OpenMPによるハードウェア動作合成システムの検討

Hardware Behavioral Synthesis System using OpenMP

中谷 嵩之† 山崎 勝弘†
Takayuki Nakatani Katsuhiro Yamazaki

1．はじめに

LSI の回路規模の増加と設計期間の短縮に伴い，回路の動作など抽象的な記述からハードウェアを自動合成する動作合成技術が実際の設計に適用されている．しかし，現在の動作合成技術では設計者の意図する並列化を行った回路を自動生成することは難しく，設計者による手動での並列化に頼っており，設計者の負担が大きい．また，並列化したハードウェアの検証においては主に RTL レベルのシミュレータなどを用いるため，検証に時間がかかるという問題もある．本研究では動作レベル記述に並列プログラミング言語 OpenMP[3]を用い，ハードウェア設計において並列動作の記述を容易にすることにより，ハードウェアの動作合成において設計者の意図した並列化手法を実現する．本稿では OpenMP を用いた動作合成システムの概要，及び FIR フィルタの並列化を例とした並列ハードウェア構成についての検討を行う．

2．OpenMP を用いた動作合成システム

2.1 OpenMP

OpenMP とは共有メモリ環境における並列処理用 API である．C++,Fortran のプログラミング言語にプラグマを追加することにより，繰り返し処理を並列化する並列リージョン，及び複数の異なる処理を並列化する並列セクションの範囲指示を行うことができる．さらに，逐次プログラムに対し，共有変数や並列動作するノード数などの指定を追加するだけで並列化が可能である．

2.2 動作合成における OpenMP の使用

図 1 に OpenMP を用いた動作合成システムの概要を示す．本手法では動作記述として OpenMP を用い，逐次プログラムの並列化を段階的に行うと共に，SMP 環境における高速シミュレーションによって並列化手法の有効性や性能の評価を行う．

並列処理の性能を検証後，動作合成を行う．既存の動作合成において，並列化は主に演算単位で局所的な処理に対して行われ[1]，データ構造やアルゴリズムに依存した大規模な並列化やパイプライン化を自動で検出することは非常に難しいが，OpenMP の並列化指示によってこれらを可能とする．また同期や並列実行時における変数へのアクセスなどを設計者が記述する必要はなく，制約に応じた最適なハードウェアの自動生成が可能である．

2.3 OpenMP の実行モデルとハードウェア

OpenMP の共有メモリモデル環境における実行時のモデルを図 2 に示す．逐次処理から実行を開始し，並列処理の開始時にスレッドを動的に生成する．処理を複数のプロセッサに割り当てて実行した後，逐次処理へ戻る．スレッドの生成に時間的コストが必要であるため，処理の粒度がある程度大きくなければ並列効果が得られない．

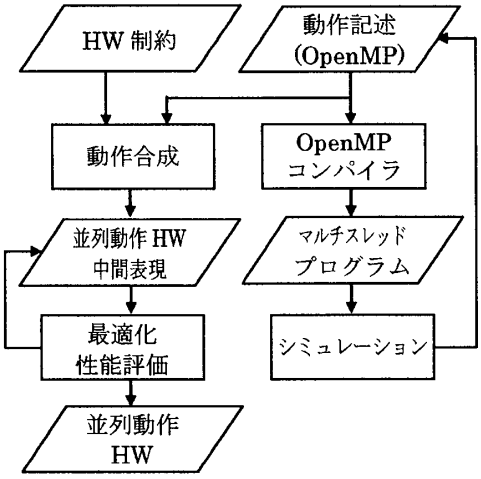

図 1：OpenMP を用いた動作合成システム

動作合成による並列動作ハードウェアは図 3 のような構成となる．並列動作に必要なハードウェアは動作合成時にあらかじめ生成し，並列処理に達すると各ハードウェアが動作を開始する．OpenMP の実行モデルと比べ，スレッド生成や引数のコピーなどのコストが必要ないため，処理の粒度に関わらず並列化効果が期待できる．

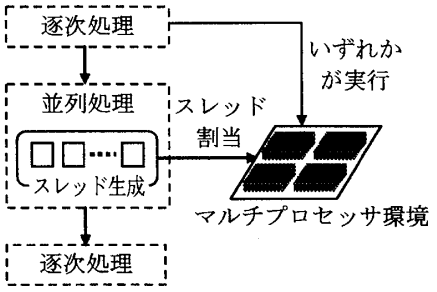

図 2：OpenMP の実行モデル

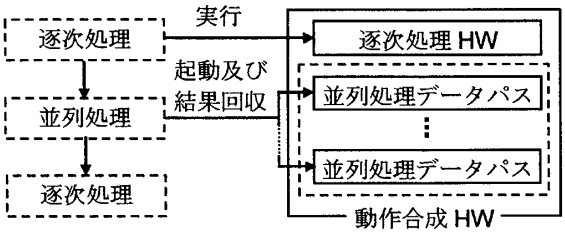

図 3：並列実行ハードウェアの動作モデル

† 立命館大学大学院 理工学研究科, Graduate School of Science and Engineering, Ritsumeikan University

3. 共有メモリモデルによるハードウェア並列化

本手法では OpenMP の実行モデルに沿ってハードウェアを生成し，その実行結果は OpenMP プログラムと等価とする．本章では OpenMP での並列化モデルとその動作合成ハードウェアの構成を FIR フィルタを例として用いる．FIR フィルタの一般的な構成は図 4 のようになり，シフトレジスタ，乗算，加算はサイクル単位で並列に動作する．

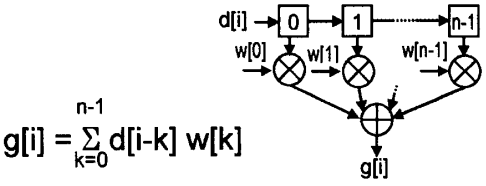

図 4：タップ数 n の FIR フィルタ

3.1. データ分割による並列化

データ分割による並列リージョンの OpenMP 記述と，並列ハードウェアの構成を図 5 に示す．データ分割では大量のデータに対する繰り返し処理を分割し，複数のノードに処理を分配する．動作合成時にノード数に応じた同一のデータパス，共有メモリへの複数同時アクセスや並列動作を制御する回路を生成する．動作を開始すると，繰り返し処理が終了するまで，制御回路はデータパスに処理を割り当てる．並列動作の制御回路はデータパスの起動や終了，必要な値の分配，結果の集約を行う．データ分割による並列化は，大量のデータに対し，同一の処理を行う画像処理や信号処理などに適している．

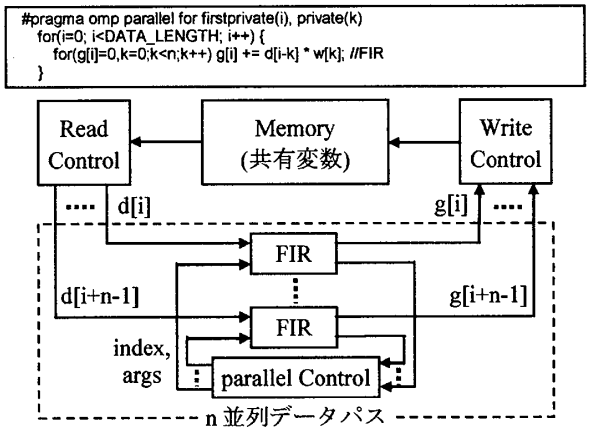

図 5：FIR のデータ分割による記述とハードウェア構成

3.2. タスク分割による並列化

タスク分割では，並列セクションにおいて同時実行可能な処理を指定，分割し，複数のノードで分配する．動作合成では並列セクション中の分割処理に対応したハードウェアを生成する．並列処理中は全ての分割処理が並列に動作し，各分割処理の終了時に他の分割処理と同期を取る．タスク分割は主に相互に依存関係のない処理を並列化する際に用いるが，依存性がある連続した処理を時系列に従って分割し，各分割処理の結果を次の分割処理の入力とすることで，パイプラインが構成可能である．

タスク分割による並列セクションの OpenMP 記述と複数のノードでの分割処理の例を図 6 に示す．タスク分割では FIR フィルタを過去の入力の保持，重みの乗算，乗算結果

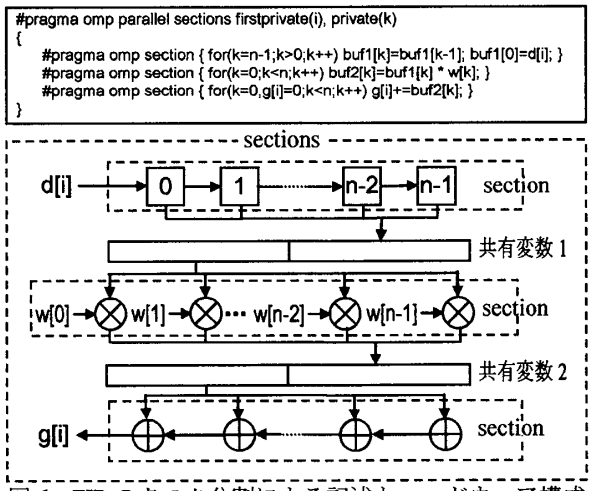

図 6：FIR のタスク分割による記述とハードウェア構成

の総和の 3 つの並列セクションに分割する．最初のセクションでは入力 d[i]を含む n 個の過去の入力を保持し，共有変数 1 に保存する．次のセクションでは共有変数 1 から値を読み出し，重みを乗算後に共有変数 2 に結果を保存する．最後のセクションでは共有変数 2 から読み出した乗算の結果の総和を計算し，g[i]を出力する．各セクションが同期を取りながら並列に動作することで，入力データに対しパイプライン処理を行う．

4. FIR の OpenMP 並列化とハードウェア最適化

FIR フィルタの並列化，及びハードウェア構成を検討するため，OpenMP プログラムを SMP 環境で並列化した．その結果，4 ノードの共有メモリ環境において，データ分割で 2.05 倍，タスク分割で 1.32 倍の性能向上が得られた．また，いずれの場合もタップ数やデータ数を増やしても速度向上が得られなかった．これは処理の粒度に対してスレッド生成によるコストが大きく，メモリアクセスにおける競合が頻発し，ボトルネックとなっているためと考えられる．本手法では，既存の動作合成やコンパイラにおける最適化に加えて OpenMP による並列化情報を用いた最適化を行う．主にメモリバンクの分割や並列ハードウェア用のキャッシュなど共有変数に対するメモリアクセスのボトルネックを解消する．

5. おわりに

本稿では OpenMP を用いた動作合成システムにおける並列化手法，及び動作合成におけるハードウェア構成について検討した．今後，提案したシステムの完成及び並列化情報を用いた最適化の実装を行い，JPEG，ウェーブレット変換，AES などに適用し，本手法の評価を行う．

参考文献

[1] 森江善之，冨山宏行，村上和彰:“ 動作合成の効率化を指向した動作レベル記述・トランスフォーメーション”，情報処理学会研究報告 Vol.2003 No.120, 2003.

[2] 岩田，田中，山崎:“ C 言語による有限ステートマシンベースのプロセッサ生成”，信学技報,VLD2001-07,ICD2001-162,FTS2001-64,pp33-38, 2001

[3] OpenMP : http://www.openmp.org/