Vol.2021-SLDM-195 No.1
2021/10/15

モンテカルロ木探索と整数線形計画法の組み合わせによる最適スケジューリング

松岡尚典[†1]　瀬戸謙修[†1]

概要： 高位合成は, LSI の大規模化と複雑化に対処する設計技術として注目されている. その 1 工程であるスケジューリングは, 生成される RTL 回路の面積や性能に大きな影響を与える. 従来手法では, 大規模な DFG に対する最適性低下や処理時間増大が問題になっていた. そこで, 効果的な発見的手法であるモンテカルロ木探索に整数線形計画法を組み合わせることで, 最適性と処理時間のバランスの取れたスケジューリングアルゴリズムを検討する.

1. はじめに

大規模集積回路(LSI: Large Scale Integrated circuits)の設計はレジスタ転送レベル(RTL: Resister Transfer Level)で詳細に記述する必要があるが, LSI の大規模化, 高集積化に伴い, 人手による RTL 記述は時間的コスト的に困難になっている. そこで注目されている技術である高位合成は, 高級言語により記述されたアルゴリズム記述から, 性能制約や面積制約を満たす回路を合成する.

高位合成の 1 工程であるスケジューリングは, アルゴリズム記述から生成された DFG に対して, 各演算の実行ステップを決定する処理であり, 実際に合成される回路の性能や面積に大きな影響を与える. 演算器の数を制約として実行サイクル数を最小化するリソース制約スケジューリング問題は, NP 困難な組み合わせ最適化問題であり, 整数線形計画法(ILP: Integer Linear Programming)を使用して定式化することで最適解を求められるが, 現実的な規模の LSI 設計に対して最適解を得ることは難しい. そのためヒューリスティック手法が用いられている. ヒューリスティック手法の 1 つとして, 効果的に最適性を高められるモンテカルロ木探索(MCTS: Monte Carlo Tree Search)が挙げられるが, 最適解に近い解を得るには探索回数を増やす必要があり, 処理時間が増大する課題がある. そこで, 小規模な問題に対して高速に最適解を生成できる ILP を MCTS ベースのスケジューリングと組み合わせることで, 最適性と処理時間のバランスの取れたスケジューリングアルゴリズムを検討する.

2. 既存手法

リストスケジューリングは, 対象 DFG の各ノードに優先順位をつけリストとして保存しておき, 優先度が高いものから順にスケジューリングを行うヒューリスティック手法である.[1] ノード数が増えた場合に優先順位の付け方のパターンが増大する上, 優先順位の付け方が結果に大きく影響するため, 最適性の保証が難しくなる.

ILP ベースのスケジューリングでは与えられた制約式の中で, 目的関数を最大化または最小化する変数の組み合わせを ILP により求める.[2] ILP はそれ自身が NP 困難な組み合わせ最適問題であり, 小規模問題は高速で最適解を生成できる一方, 最適性を維持しつつ大規模な問題に適用するのが難しい.

MCTS スケジューリングは, モンテカルロ法[3]に基づいたヒューリスティック手法であり, ランダムプレイアウトにより評価値を得て探索木を構成する. 評価関数による推測評価値ではなく, 実際にプレイアウトを行った結果から評価値を得るため, 局所的な評価が全体評価に影響を与え難いリソース制約スケジューリングにおいて有効な手法である. 探索を進める際に過去の評価から最適と推測されるものを選ぶのか, それとも探索回数が少ないものを冒険的に選択するのかのバランスを取ることが難しく, 様々な手法が提案されている. 代表的なものに多腕バンディッド問題で扱われる UCB(Upper Confidence Bounds)を木構造に適用した UCT(Upper Confidence Bounds applied to Trees)がある[3].

3. MCTS スケジューリングと ILP スケジューリングの組み合わせ方の検討

提案手法の考え方として, ノード数の大きな問題に対して, まず MCTS スケジューリングを行い, 残りのノード数が ILP で高速に解ける規模になったら, ILP スケジューリングに移行して高速に最適解を求める方針である.

MCTS と同程度の処理時間で, ILP で最適解を求められる範囲を調査するため, まず, 各リソース数を 1 とした時のリソース制約スケジューリングにおいて, ILP と MCTS によるスケジューリングの処理時間と解の品質を評価する実験を行った. MCTS によるスケジューリングでは乱数を用いるため, 100 回行った際の平均値を結果として示す. MCTS の探索回数は 500 回とし, ILP ソルバには Gurobi Optimizer を用いた.

図 1 に示すようにノード数 40 個程度で, ILP が MCTS と比べて処理時間が長くなることがわかった. また, 図 2 より, MCTS によるリソース制約スケジューリングでは, 32 ノードの小規模問題においても最適解と比べて平均 3 サイクル

1 東京都市大学
Tokyo City University

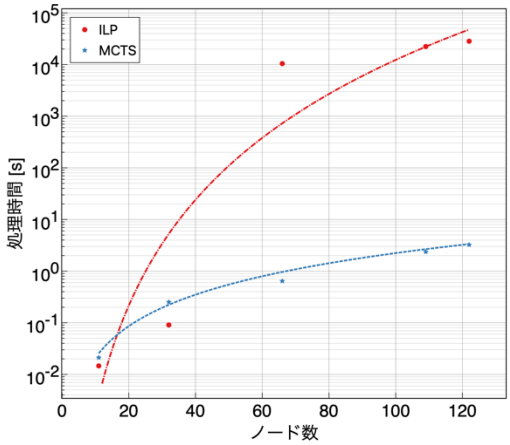

図 1. リソース制約スケジューリングにおける ILP と MCTS の処理時間比較

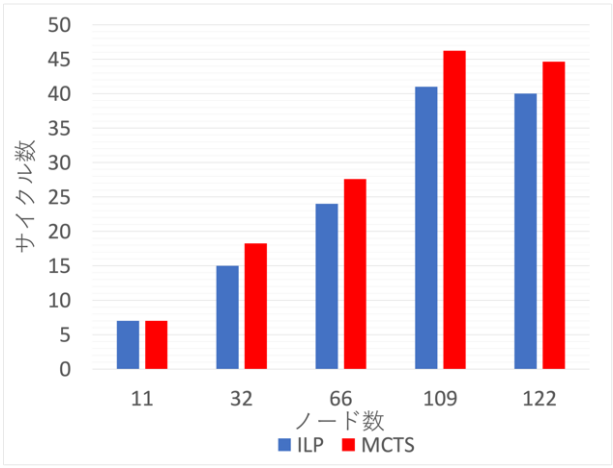

図 2. リソース制約スケジューリングにおける ILP と MCTS のサイクル数比較

悪化していることがわかった. そこで, MCTS と ILP どちらも同程度の処理時間でスケジューリングを行えるノード数を 32 と設定し, MCTS スケジューリングの残りのノード数(未スケジュールのノード数)が 32 個になった段階で ILP に移行する形で, MCTS と ILP を組み合わせた.

ILP と MCTS によるスケジューリングの処理時間と解の品質評価時と同様に, MCTS の探索回数は 500 回, ILP ソルバに Gurobi Optimizer を用い, 提案手法を 100 回実行した結果の平均値を図 3, 4 に示す. 122 ノード時において, 提案手法の要する処理時間は MCTS の 2 倍程度であり, ILP と比較した際に処理時間の急激な増大は抑えられている(図 3). また全過程が ILP で行われる 32 ノード時を除き, 図 4 に示すように最適解 24 サイクルの 66 ノード時で, 純粋な MCTS 解 27.6 サイクルに対して提案手法 26.49 サイクルと, 最大 31%の改善が見られた. 一方で, ノード数の増大に伴い MCTS 解の改善率は低下しており, 最適解が 40 サイクルの 122 ノード時には, 純粋な MCTS 解 44.63 サイクルに対して提案手法 44.25 サイクルと, 8.2%にとどまる結果となった.

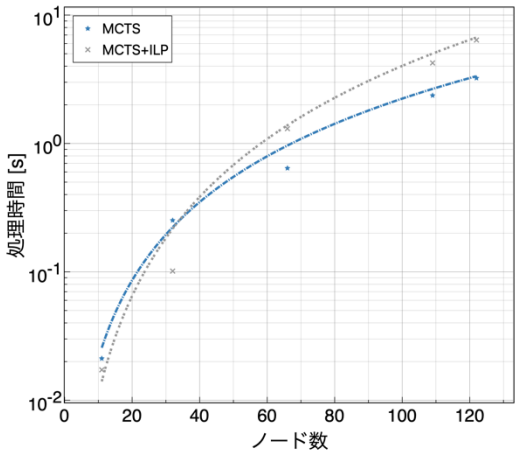

図 3. リソース制約スケジューリングにおける MCTS と提案手法の処理時間比較

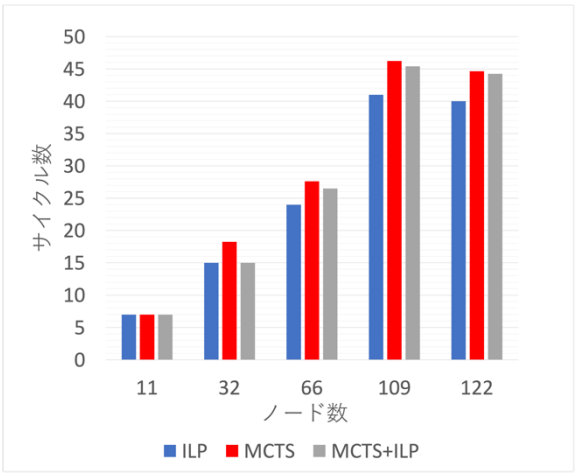

図 4. リソース制約スケジューリングにおける ILP と MCTS, 提案手法のサイクル数比較

4. まとめ

リソース制約スケジューリングにおいて, 提案する MCTS と ILP を組み合わせたスケジューリング手法を用いることで, MCTS のみを用いる場合と比べてもサイクル数が削減できた. しかし, ノード数の増大に伴い, ILP ベースの手法と比べて, サイクル数が悪化している. これはヒューリスティック手法である MCTS を使用しているためである. 提案手法により生成する解を最適解に近づけるためには, 一部の範囲のみの最適解を求めるだけではなく, 複数の範囲において最適解を求める必要があると考えられる.

参考文献

1) Keith D. Cooper, Philip J. Schielke, and Devika Subramanian. (1998). An Experimental Evaluation of List Scheduling. Tech. rep., Deptartment of Computer Science, Rice University.
2) Giovanni De Micheli. (1994). Synthesis and Optimization of Digital Circuits. McGraw-Hill Higher Education. pp. 198–202.
3) L. Kocsis and Cs. Szepesvari. (2006). Bandit based monte-carlo planning. in European conference on machine learning. Springer, pp. 282–293.