FPGA を対象としたデータ駆動型プロセッサの設計自動化フローの検討

1235066　長野 寛司　【 コンピュータ構成学研究室 】

A Study on FPGA Circuit Design Automation Flow for Data-Driven Processor

1235066　Kanji Nagano　【 Advanced Computer Engineering Lab. 】

1　はじめに

近年，IoT(Internet of Things) 技術の普及により，エッジ機器の処理負荷が増大している．それに伴って，高性能化、低消費電力化、多様化の需要が高まっている．これに対し，データ駆動型プロセッサ (DDP:Data-Driven Processor)[1] は，低消費電力で動作し，異なるデータ流を多重に処理可能で高性能なため，有効なアーキテクチャである．また，FPGA は応用に合わせた柔軟な回路設計が可能で，多様性が求められるエッジ機器に有効である．以上から，DDP を FPGA 上に実装して IoT エッジ機器を実現する方法が有望である．しかし，既存の FPGA と回路設計ツールは同期式回路に最適化され，非同期式回路の DDP を最適に設計することが困難である．これに対して，既存の回路設計ツール (Intel 社 Quartus) の機能を巧妙に使い，手動で最適な DDP を設計する手法を提案した [2]．しかし，この設計手法には，多大な設計時間が必要であり，また回路の動作保証が十分ではなかった．

本研究では，簡易的にかつ動作が保証される設計を行えるように，FPGA を対象とした DDP の設計自動化フローの提案をする．

2　DDP の設計自動化の課題

DDP の各ステージは図 1 の様に STP(Self-Timed Pipeline) により構成されている．各ステージはタイミング制御回路の C_{i-1}(C 素子) から CP_{i-1} が立ち上がり，レジスタの DL_{i-1}(Data Latch) が解放され，後段ステージへデータを送信する．C_{i-1} と C_i がハンドシェイク通信を行うことで，C_i の CP_i が立ち上がる．LUT は 1 か 0 の値を一時的に保存しておく RS フリップフロップレジスタの役割を持っている回路である．

DDP の設計課題であるタイミング制約，配置・配線について述べたのち設計自動化の課題を述べる．

2.1　タイミング制約

回路を正常に動作させるためにタイミング制約を満たす必要があり，通常の C 素子が制約を満たしているかは，筑波大学のタイミング検証ツール [3] を用いることで検証できる．しかし，DDP には通常の C 素子から派生させたタイミング制御回路があるため，それらに応用する方法が必要である．

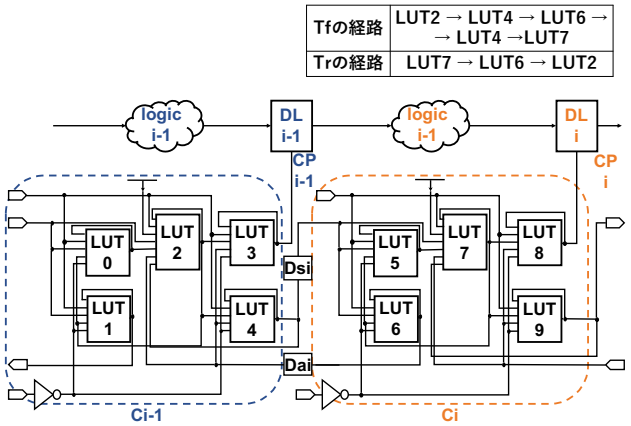

図 1　STP ステージの構成

LUT は RS フリップフロップと同等の機能をもち，本来回路記述に不要なラッチを追加して，Quartus にレジスタとして認識させている回路である．レジスタ間のタイミングは Quartus を用いて抽出できるため，LUT 間のタイミング情報を抽出して最適な遅延回路を設計できる．

DL_{i-1} からデータが解放され，DL_i に書き込まれる前に $logic_i$ の処理が終了する必要があるセットアップ時間制約は，遅延回路 Ds_i でタイミング Tf を調整することで制約を満たす．DL_i にデータが書き込まれた後の一定時間，書き込まれるデータが安定している必要があるホールド時間制約は，遅延回路 Da_i でタイミング Tr を調整することで制約を満たす．

2.2　配置・配線

FPGA は配置・配線による遅延の影響が ASIC と比べて大きいため，回路を凝集して配置・配線することで配線遅延を削減することが望ましい．また，IoT 向けエッジ機器のマルチコア構成に拡張可能な設計にするために，再配置可能な配置・配線案が必要である．

2.3 設計自動化の課題

設計を自動化するためには，全ての作業を CUI 上で実行する必要がある．そのためには，Quartus の GUI 作業に対応した Tcl スクリプトや，タイミング検証 [3] で使用する JSON ファイルなどを自動生成する必要がある．

3 DDP の設計自動化フロー

自動化に必要なファイルや Quartus の配置配線結果レポート (fit.rpt) 等を Perl スクリプトで処理し，Quartus の GUI 作業と等価な Tcl スクリプトやタイミング解析・検証に用いるファイル等を出力し，CUI(NiosII Command Shell) で動作させる．

3.1 回路設計プロジェクト作成

Verilog ソースコード，設計対象 FPGA のデフォルトのプロジェクト設定ファイル (qsf ファイル)，Quartus のバージョン，プロジェクト名を入力とし，新たなプロジェクト qsf ファイルを作成してプロジェクトを生成する．ディレクトリから対象ファイルを自動的にプロジェクトに追加することで，ファイルの追加漏れを防ぐ．

3.2 パーティションの設定

設計する回路が合成時に最適化されないようにパーティションを設定する．最適化の対象によって正しくタイミング解析・検証ができなくなる事象が発生するためこれを防ぐ．配置配線結果レポート (fit.rpt) からパーティション設定に必要な情報を抽出して Tcl スクリプトを生成する．

3.3 Region 設定・配置・配線

DDP 全体の回路を凝集して合成して配線遅延を削減するために Region の設定を行う．配置配線結果レポート (fit.rpt) から，Region 設定に必要な情報を抽出し，Tcl スクリプトを生成する．配置・配線は Quartus の自動配置・配線機能を実行する．

3.4 タイミング解析・検証

現在の遅延回路でタイミング制約を満たしているかを確認する．そのために，タイミング解析に使用する Tcl スクリプト，SDC（設計制約）ファイルを生成してタイミング解析を行う．その後，タイミング検証に用いるパスの情報を JSON 形式で出力し，タイミング解析結果と併せてタイミング検証ツール [3] へ入力する．タイミング制約を満たしていなければ，設計者が Verilog ソースコードを書き換えてタイミングを調整する．

3.5 コンフィグレーション

タイミング検証で違反がないことを確認後，プログラムファイルの sof ファイルを入力として，FPGA へコンフィグレーションを行う Tcl スクリプトを出力し，FPGA へ回路情報を書き込む．

表 1　評価結果

	従来手法 [2]	提案手法
設計時間 [min.]	324	237
DDP 回路規模 [LEs]	4830	4893
DDP 回路面積 [W*H]	25*16	20*28
スループット [packets/s]	31.4M	30.8M

4 評価

提案手法の評価のために使用した FPGA ボードは Intel 社製 MAX10 DE10-LITE である．回路設計ツールは Intel 社製 Quartus Standard Edtion 18.0 を用いた．評価指標は，DDP の設計時間，DDP の回路規模，スループットである．評価対象は [2] の従来手法で設計した DDP とした．設計時間は回路設計プロジェクト作成からタイミング調整終了までとする．

設計時間は 27%削減され，回路規模は 1.3%，回路面積は 47%増加し，スループットは 2%低下した．回路の性能が低下しているように感じられる結果だが，従来手法 [2] で設計した DDP をタイミング検証してみると，タイミング制約を満たしていなかった．提案手法はタイミング制約を満たし，また設計時間も削減できているため，有効な手法であるといえる．

5 まとめ

本研究では，FPGA を対象として，動作が保証た DDP の設計自動化フローを検討した．

Quartus の GUI 処理を CUI で実行する Tcl スクリプトや，タイミング検証ツール [3] へ入力するファイル等を Perl で出力することで，DDP の設計を自動化できる環境を構築できた．

今回の配置・配線案は Quartus の自動配置・配線機能に任せており，各モジュールが DDP 全体の Region 内のどこに配置・配線されるか分からず，遅延回路を調整して再合成する度に変更され，遅延も微妙に変化する．[2] の様に設計者の配置・配線案を適用させることができれば，マルチコア化を考えた際に全てのコアを同等の性能で設計できるが，これに関しては今後の課題である．

参考文献

[1] H. Terada, et al., “DDMP’S: Self-Timed Super-Pipelined Data-Driven Multimedia Processors,” Proc. IEEE, Vol. 87, No. 2, pp. 282–296, Feb. 1999.

[2] K. Nagano, et al., “Area Afficiant Implementation of Data-Driven Processors,” ISFT2019, pp. 58, Aug. 2019.

[3] S. Yoshikawa, et al., “Pipeline Stage Level Simulation Method for Self-Timed Data-Driven Processor on FPGA,” iEECON, pp. 1–5, Mar. 2020.