Vol.2019-EMB-51 No.5
2019/6/21

コンテキスト指向プログラミングによる FPGA 動的部分再構成の考察

佐藤未来子[†1]　渡辺晴美[†1]　谷川郁太[†2]
大川猛[†1]　小倉信彦久[†3]　久住憲嗣[†2]

概要：本研究では，コンテキスト指向プログラミング（COP）のレイヤ切り替えに応じて，FPGA を動的に部分再構成可能なシステムについて述べる．近年，自動運転技術や工場ラインの自動化などシステムの複雑化や高性能化が顕著となり，FPGA の導入が注目されている．FPGA は回路全体の再構成だけではなく，回路全体を停止することなく一部の回路のみを動的に部分再構成することが可能である．本研究では，FPGA 動的部分再構成の機能に着目し，本研究の基盤としている RTCOP(RoboT Context-Oriented Programming)のレイヤ管理において，ソフトウェアモジュールの切り替えと共に，FGPA を動的に部分再構成するようメソッドを管理する．本稿では，COP における FPGA の動的部分再構成の課題と，RTCOP においてコンテキストに応じて FPGA を動的に再構成する仕組みについて述べる．

キーワード：コンテキスト指向プログラミング，FPGA，動的部分再構成

1. はじめに

デジタルトランスフォーメーション（DX）や IoT の時代となり，システムはより複雑で高性能なものが要求されている．近年の生産工場では既に多くのロボットを採用して工場の自動化を進めている．また，次世代の自動運転技術では，運転方法を環境に応じて変更するなど，周囲の状況や要求に応じた複雑なサービスを提供する必要がある．このようなシステムにおいて，FPGA を併用することによる，エッジコンピューティングの実時間処理の高速化が期待されている．例えば，ロボットにおける，画像認識処理や視覚情報に基づく処理など，高速化が要求されている処理に対して FPGA を活用することは有効である．

これまで本研究では，高性能かつ複雑なシステムを実現するために，(1) 横断的設計を要する問題，(2) 高性能化，(3) 実行時のシステム変更への対応，などの課題を解決するために，コンテキスト指向プログラミング（Context-Oriented Programming: COP）に着目している[1][2]．FPGA との協調設計に関しては，アスペクト指向プログラミングにおける協調設計[3][4]は提案されているものの，COP では本研究以外では提案はされていない．

本稿では，FPGA を含む COP におけるシステム開発手法について述べる．COP はコンテキストに依存した振る舞いをレイヤとしてモジュール化し，それらを実行時のコンテキストの変化に応じて切り替える．本研究では COP のレイヤ定義の中に従来のソフトウェアのメソッドと共に，ハードウェア回路を含めた横断的設計を可能にする．以下，既発表の研究である RTCOP (RoboT Context-Oriented Programming) [2]において FPGA の動的再構成を行う際の課題と，COP のレイヤ管理において，コンテキストに応じて FPGA を動的に再構成する仕組みについて述べる．

†1 東海大学
†2 九州大学
†3 東京都市大学

2. COP における FPGA 動的再構成の課題

FPGA の再構成にレイヤアクティベーションの概念を適用し，コンテキストに応じた FPGA のアクセラレーション回路を動的に再構成する場合，(1)レイヤのアクティベーション時にどのレイヤに対して，どのハードウェア回路を再構成するかというマッピングの指定方法を追加すること，(2)FPGA の動的再構成にかかる時間はソフトウェアのメソッド切り替えに比べてはるかに長いため，レイヤアクティベーション時のソフトウェアとハードウェアの同期方法を検討すること，などが課題となる．課題(1)については，RTCOP におけるハードウェアメソッドの定義方法と，その処理方法を検討した．また，課題(2)については，同期方法を検討するための予備実験として，FPGA 評価ボードを用いた RTCOP において，レイヤアクティベーション時の FPGA の動的再構成の基本性能を評価している．

3. RTCOP におけるハードウェア回路の動的再構成方法

RTCOP で提案している FPGA を扱うためのフレームワークを図 1 に示す．本フレームワークのレイヤ管理ユニット（Layer Management Unit: LMU）がレイヤアクティベーションの要求を処理する．RTCOP のソフトウェアメソッドの切り替えは，COP レイヤをアクティベートした時に，LMU が仮想関数テーブルを書き換えることにより実現している[2]．これに対して，ハードウェア回路の切り替えは，レイヤ定義のメソッドにハードウェア回路モジュールを対応付けておき，レイヤのアクティベート時に FPGA の動的再構成を実施する．近年，ベンダーの高位合成（High-Level Synthesis: HLS）ツール[5][6]が標準化されており，RTCOP のフレームワークにおいても，HLS を利用したハードウェア回路の生成，書き換え，CPU 間データ送受信を行う設計としている．

RTCOP のレイヤ定義に紐づけされたハードウェア回路

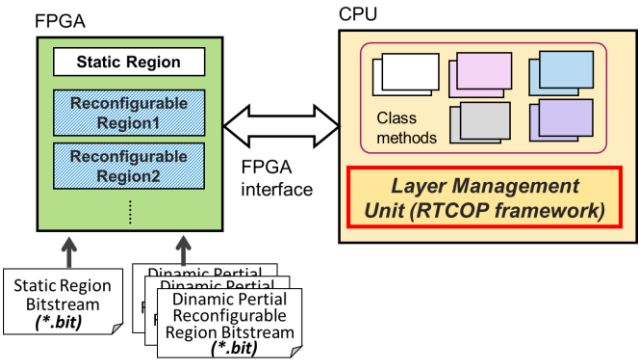

図 1　RTCOP における FPGA フレームワーク

の実体は，HLS により動的部分再構成用ビットストリームとして生成される．動的部分再構成用ビットストリームを FPGA 上に配置する際の領域情報などは，動的部分再構成のハードウェア回路のコンパイル時の配置配線段階でビットストリームに書きこまれる．そのため，LMU においてレイヤとビットストリームとの対応関係を管理しておくことにより，アクティベート時に，レイヤに応じて指定されたビットストリームが所定の領域へ書き込まれる．なお，クロック，リセット制御などの基本的な回路は，本フレームワークによりビットストリームを生成しておき，システム起動時に FPGA の静的領域へ書き込む．

図 2 に，RTCOP におけるプログラム開発フローの概要を示す．RTCOP のアプリケーション開発者は，レイヤ定義のコード（ファイル拡張子は「lcpp」と「lh」）[2]と，ハードウェア回路の再構成用コンフィグレーションファイル（XML 形式ファイル）を作成する．RTCOP の「レイヤコンパイラ」は，上記 2 種類のファイルから，CPU で実行するソフトウェアメソッドと，FPGA 上で実行するハードウェアメソッドを生成する．その後，CPU 向けに実行ファイルを生成し，FPGA へ書き込むためのビットストリームファイルを，FPGA ベンダツールを使用して生成する．

図 3 に，動的部分再構成用のビットストリームにレイヤメソッドを対応付けるためのソースコードとコンフィグレーションファイルの例を示す．コンフィグレーションファイルには，どのレイヤがアクティベーションされたときに，どのクラスのどのメソッドを，どの領域で再構成するのかということを記す．本コンフィグレーションファイルとレイヤ定義から，アプリケーションプログラムとビットストリームを生成し，LMU がアクティベーション要求を処理し，FPGA 用インタフェースを介して FPGA を再設定する．

4. おわりに

本稿では，RTCOP(RoboT Context-Oriented Programming) のレイヤ管理において，ソフトウェアモジュールとの協調設計により，FGPA の動的に部分再構成を行う方法について述べた．今後は，FPGA 評価ボードを用いた，RTCOP における FPGA の動的部分再構成の基本性能の結果を踏まえ，

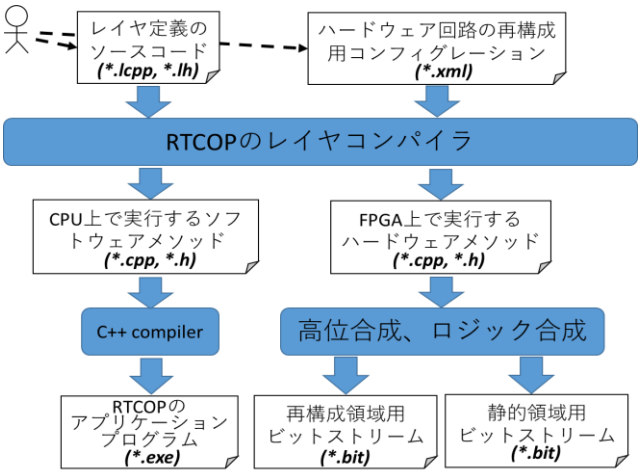

図 2　RTCOP におけるプログラム開発フロー

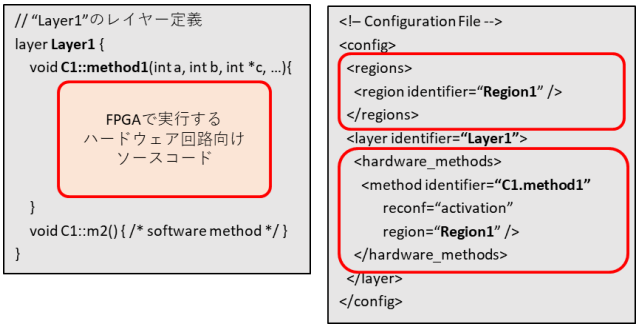

図 3　RTCOP における FPGA 向けレイヤ定義とコンフィグレーションファイルの記述例

RTCOP レイヤ切り替えにおけるソフトウェアメソッドと FPGA 動的部分再構成との同期方法を検討する必要がある．

参考文献

[1] H. Watanabe, M. Sugaya, I. Tanigawa, N. Ogura, and K. Hisazumi, "A Study of Context-Oriented Programming for Applying to Robot Development," Proceedings of the Workshop on Context-oriented Programming (COP) 2015, ECOOP 2015, 2015.

[2] Ikuta Tanigawa, Kenji Hisazumi, Nobuhiko Ogura, Midori Sugaya, Harumi Watanabe, and Akira Fukuda：RTCOP: Context-Oriented Programming Framework based on C++ for Application in Embedded Software, Second Int. Conf. on Information Science and System (ICISS 2019), 2019.

[3] João M.P. Cardoso, Tiago Carvalho, José G.F. Coutinho, Wayne Luk, Ricardo Nobre, Pedro Diniz, and Zlatko Petrov. 2012. LARA: an aspect-oriented programming language for embedded systems. In Proceedings of the 11th annual international conference on Aspect-oriented Software Development (AOSD '12). ACM, New York, NY, USA, 179-190, 2012.

[4] Marco Aurélio Wehrmeister and Marcela Leite. 2014. On Generating VHDL Descriptions from Aspect-Oriented UML/MARTE Models. In Proceedings of the 2014 Brazilian Symposium on Computing Systems Engineering (SBESC '14). IEEE Computer Society, Washington, DC, USA, 67-72, 2014.

[5] Intel Corp., "Intel® HLS Compiler: Fast Design, Coding, and Hardware," White Paper, 2018.

[6] Xilinx Inc., "Vivado Design Suite User Guide," UG902 (v2018.3) December 20, 2018.