マルチ FPGA におけるスケジュールシステムの提案

LI YANZHI[1]　石綿陽一[2]　菅谷みどり[3]

概要： FPGA をクラウドでの応用は性能とクライアント側のエネルギコストにバランスをとる点から注目されている。この際、複数のユーザを安全で効率的に FPGA を共有することが必須となる。本提案では、Flow-in-Cloud(FiC)というマルチ FPGA システムのタスクスケジュールシステムを提案する。このシステムは、ユーザの要求に応じて、各 FPGA の設定状況を考慮し、タスクをスケジュールするシステムである。複数のユーザが FiC 内の FPGA を使うとき、全体的に応答性を向上させることを目指している。

キーワード：マルチ FPGA、資源管理、スケジューラ

Keywords: Multi FPGA, Resource Manager, Scheduler

1. はじめに

近年、ロボット上で画像認識や SLAM（Simultaneous Location and Mapping）などの高負荷なアルゴリズムを実装し、FPGA（Field Programable Gate Array）の計算力により性能向上させる研究が提案されている [1][2]。しかし高性能な FPGA は高価で、処理コストも高い。こうした FPGA を実装するのは、費用や、バッテリー駆動による電力消費の問題がある。そこで、高性能な FPGA を必要に応じて、クラウドにおける利用するための研究が、様々な研究者によりなされている。例えば、Microsoft 社は FPGA の仮想化により、高性能な FPGA を利用可能としたが、仮想 FPGA がクラッシュするリスクは高い課題がある[3]。また Ahmed らは [4]、FPGA のロジックをカプセル化して、必要に応じて各ユーザに割り当てるマルチテナント方式で FPGA を複数のユーザに利用できる方法を示した．しかし、どんな高性能な FPGA を利用する場合でも、すべてのユーザが使える資源は限界があるという欠点が避けられない課題がある。これらの研究に対して、データの流れという視点から、Flow-in-Cloud（以降 FiC）というシステムが提案された[5]。複数の FPGA リソースを複数のクライアントから利用でき、かつ複数の FPGA を複数ロボットに使うシステムを実現できる可能性があることから、近年注目されている。

2. Flow-in-Cloud (FiC)の概要

FiC は、データの流れという視点から、複数の FPGA を回路交換ネットワークでつなぎ、パイプラインの形でデータを処理するシステムである[5][6]。図 1 に FiC 構成を示した。FiC は FiC-SW（FPGA Switching node of the FiC platform）という FPGA ノードを 中央の回路交換ネットワーク（Circuit Switching Network）により接続する形式の FPGA ネットワークである。FPGA-SW というノードの中には大まかに二つの領域がある。一つは、ボードを相互に接続する STDM(Static Time Division Multiplexing)スイッチを含むスタティック領域である。もう一つは、高位合成（High-level synthesis 以降は HLS)モジュールを実装する部分再構成（Partial Reconfiguration）可能な領域である。実際のアプリケーションを動作させる場合は、この再構成可能な部分だけを動的に再構成できる。

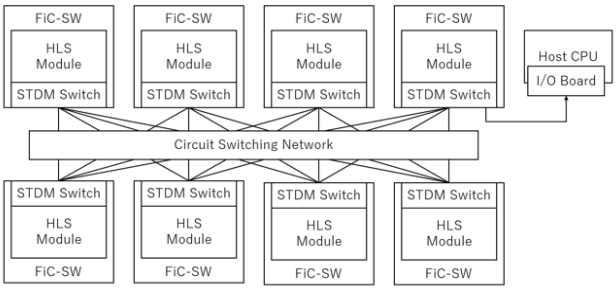

図 1　FiC システムの構成

各ボード間は、高速交換ネットワークでデータを直接転送できる。各ボードが持つ HLS モジュールをネットワークで相互に接続し、パイプラインのようにしてデータを処理することができる。

山倉らは、各ボードの上に ficwww という、マネージメントシステムのプログラムを設計した[7]。山倉らは、ficwww が提供した API を通して、FPGA をコンフィグレーションなどの操作を行えるようにした。また、複数の FPGA ボードの管理を行う FiC-RM（Resource Manager）を実装した。FiC-RM は連結している FPGA の実行環境をスライスという単位に分け、ユーザの要求に基づき、スライスを確報、開放、アプリケーションの読み込むなどを行う仕組みである。

3. 動的再構成のための FiC の課題

2 節に示したように FiC は、複数の FPGA が持つシステムであり、パイプラインの形でデータを処理し、一つの FPGA が持つ計算力を超えて、複数のユーザに使える利点がある。しかし、実際の応用については、まだ課題がある。

一つ目は、FPGA の再構成と CPU プログラムの切り替えと違い、切り替え時間が長い点である。これは FPGA の特性による問題なので、再構成の時間を短くするのは難しい。

さらに、再構成の時点をきちんと決めなければ、システムが遅くなる可能性がある。この問題を解決し、応答性をあげるためには、CPU でのスケジューラと違い、再構成を考慮したアルゴリズムを検討する必要がある。二つ目は、現状 FiC の実装についての問題である。現状の FiC は、ネットワークを経由、JSON ファイルの形で HLS コードを転送する形式で実装されている。これは、ネットワーク状況によって、転送時間が長くなる可能性があり、システムの応答性が低くなる問題がある。問題を解決するためには転送するファイルのサイズを抑えるために、HLS コードのキャッシュの実装を考える必要がある。

4. FiC のスケジュールシステムの提案

本提案は、先に述べた 2 つの課題である切り替えと実装の問題を解決する。具体的には、FPGA ボードの割り当て手法として、タスクバッファを用いたスケジューリングアルゴリズムを含む、HLS モジュールのキャッシュとスケジュール方法を提案する。まずはいくつの用語を定義する。

1. ユーザ：FiC での計算を要求するエンティティ。人に限らず、ロボットやセンサーノードなどもユーザになれる。
2. アプリケーション：ある目的のため FiC で実行する HLS モジュールである。複数の HLS モジュールを一つのフローになる場合は、一つのアプリケーションと考える。
3. データ：ここでは、アプリケーションに入力や出力するデータを示す。
4. タスク：特定のアプリケーションに特定のデータを入力して、出力データを求める過程。アプリケーションや入力するデータが明白にするまで、タスクになれない。

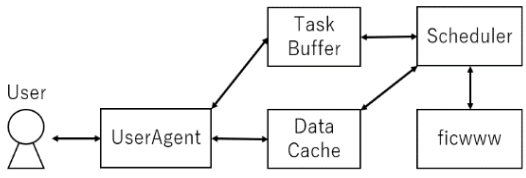

図 2　スケジュールシステムの構成

図 2 はこのシステムの構成を示す。ユーザはユーザエージェントが提供した API を呼び出して、HLS モジュールとデータを別々にアップロードして、タスクの実行を要求する。スケジューラはこの要求に応じて、ficwww を経由、スライスを割り当て、FPGA を構成して、タスクを実行させる。アプリケーションをアップロードするときはキャッシュに保存するだけ、実際にタスクを実行させる直前に、FPGA を再構成する。また、タスクが実行しても、HLS モジュールをこのままキャッシュに残り、次のタスクにも使えるように、実装の問題を解決する。

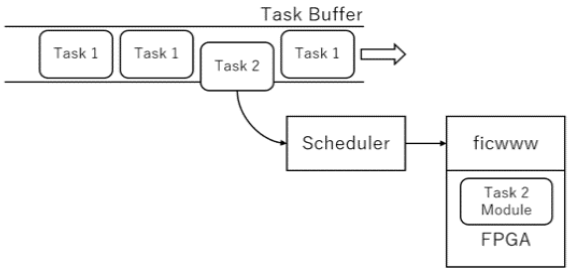

図 3　スケジュールのアルゴリズム

すべてのタスクは、タスクバッファに保存し、スケジューラがこのバッファーの中から、一つのタスクを選んで実行させる。タスクを選ぶのは、FPGA を HLS モジュールで再構成する時間やこのタスクの実行がかかる時間などを統合して、切り替えるにおける問題をできるだけならないように、次の実行するタスクを選ぶ。図 3 は、FPGA がすでに Task 2 のモジュールで構成されたとき、Task 1 の待ち時間が長くなかった限り、Task 2 をより高い優先順位にして、先に実行するスケジュール方法のイメージを示した。

5. おわりに

本論文は，FiC というシステムを述べ、このシステムを複数のユーザに使うためのスケジュールシステムを提案しました。今後の課題として、このシステムの実装と評価、さらに応答性の視点からスケジューラを最適化していく。

参考文献

[1] M. Abouzahir, A. Elouardi, S. Bouaziz, O. Hammami and I. Ali, "High-level synthesis for FPGA design based-SLAM application," 2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA), pp. 1-8, (2016).

[2] Y. Nitta, S. Tamura and H. Takase, "ZytleBot: FPGA Integrated Development Platform for ROS Based Autonomous Mobile Robot," 2019 29th International Conference on Field Programmable Logic and Applications (FPL), Barcelona, Spain, pp. 422-423, (2019)

[3] A. Putnam et al., "A reconfigurable fabric for accelerating large-scale datacenter services", Proc. Int. Symp. Comput. Archit., pp. 13-24, (2014)

[4] Ahmed Khawaja and Joshua Landgraf and Rohith Prakash and Michael Wei and Eric Schkufza and Christopher J. Rossbach, "Sharing, Protection, and Compatibility for Reconfigurable Fabric with AmorphOS", 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18), pp. 107-127 (2018)

[5] Musha, K., Kudoh, T., & Amano, H. (2018). “Deep learning on high performance FPGA switching boards: Flow-in-cloud.” Applied Reconfigurable Computing: Architectures, Tools, and Applications - 14th International Symposium, ARC 2018, Proceedings , pp. 43-54 (2018).

[6] K. Hironaka, B. Akram and H. Amano: “Multi-FPGA management on flow-in-cloud prototype system”, 20th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD 2019), pp. 443–448 (2019).

[7] 山倉　美穂. 高野　了成. Akram Ben Ahmed. 天野　英晴: "マルチ FPGA クラウドシステムにおけるマルチテナント式資源管理の開発", 信学技報, vol. 120, no. 168, RECONF2020-21, pp. 13-18, (2020)