FPGA 間通信フレームワーク CIRCUS を利用した 複数 FPGA によるグラフ幅優先探索の提案

溝谷　祐大† 　小林　諒平†† 　藤田　典久†† 　朴　 泰祐†† 　天笠　俊之††

† 筑波大学大学院理工情報生命学術院　〒 305–8577 茨城県つくば市天王台 1 丁目 1-1
†† 筑波大学 計算科学研究センター　〒 305–8577 茨城県つくば市天王台 1 丁目 1-1
E-mail: †mizotani@kde.cs.tsukuba.ac.jp, ††{kobayashi,amagasa}@cs.tsukuba.ac.jp,
††{fujita,taisuke}@ccs.tsukuba.ac.jp

あらまし　グラフ構造は，様々なデータをノードとエッジで表したデータ構造のことであり，我々の身の回りの多種多様なデータの関係性を表すのに有用である．グラフの分析は盛んに行われており，グラフから様々な情報が取得されている．グラフの分析アルゴリズムの中でも，幅優先探索は最も広く使われているアルゴリズムである．幅優先探索とはグラフ探索アルゴリズムの一種であり，デジタル回路のテスト・検証，道路ネットワークの解析など，幅広い分野で応用されている．しかし，近年グラフの大規模化によって，幅優先探索に多大な計算コストが必要となることが多い．また，不規則なメモリアクセスが多くなるためメモリ帯域を有効に利用できないといった問題がある．ここで我々は FPGA に着目した．FPGA とは，任意の回路をプログラミングによって繰り返し実装可能なハードウェアチップである．その性能上の特徴は各回路の並列性を利用した並列度の高い処理が可能なことである．また，FPGA では外部通信用光リンクを利用できる．この外部通信用光リンクは FPGA 上の回路と直接接続されているため超低レイテンシで他の FPGA と通信することが可能となる．この特徴を活用する技術として FPGA 間通信フレームワーク，CIRCUS がある．本研究では，CIRCUS を利用し，複数 FPGA を使い幅優先探索を実装する．

キーワード　FPGA, OpenCL，幅優先探索, ハードウェアアクセラレータ, グラフデータベース

1 はじめに

グラフ構造は，様々なデータをノードとエッジで表したデータ構造のことであり，我々の身の回りの多種多様なデータの関係性を表すのに有用である．グラフの分析は盛んに行われており，グラフから様々な情報が取得されている．

グラフの分析アルゴリズムの中でも，幅優先探索は最も広く使われているアルゴリズムである．幅優先探索とはグラフ探索アルゴリズムの一種であり，デジタル回路のテスト・検証，道路ネットワークの解析など，幅広い分野で応用されている．幅優先探索では，根ノードから隣接した全てのノードを探索し，そこからさらに隣接するノードに対して同様のことを繰り返して探索対象ノードを見つけることができる．

しかし，近年グラフの大規模化によって，幅優先探索の対象となるグラフも大規模化が予想され，幅優先探索において多大な計算コストを要することが推測される．このような大規模なデータを処理するために GPU や FPGA といったハードウェアアクセラレータの利用が注目されている．

FPGA とは，任意の回路をプログラミングによって繰り返し実装可能なハードウェアチップであり，任意の回路をプログラミングによって繰り返し実装可能となっている．FPGA 上には，Block RAM (BRAM) という FPGA チップ上の回路が 1 サイクル以内にアクセス可能であり，非常に高速なメモリが存在する．しかし，通常 BRAM は容量が小さい．そのため，FPGA を利用する際，BRAM 容量を節約することや，データの一部を低速なグローバルメモリに保持することが必要となる．そこで，複数 FPGA を用いることで物理的に利用できる BRAM の容量を増やし，処理を高速化する動きがある．

FPGA 間を通信する技術として，FPGA 間通信フレームワークがある．FPGA 間通信フレームワークとは，FPGA の外部通信用光リンクを利用でき，この外部通信用光リンクは FPGA 上の回路と直接接続されているため超低レイテンシで他の FPGA と通信することが可能という特徴を活用した技術である．

これらの点を踏まえ，我々は複数 FPGA を利用した幅優先探索の実装を提案する．FPGA 間通信フレームワークの CIRCUS [1] を利用した複数 FPGA を用いた幅優先探索の手法を考案し，実装を行った．

2 前提知識

2.1 幅優先探索

幅優先探索とは，グラフ探索アルゴリズムの一種である．根ノードから隣接した全てのノードを探索し，そこからさらに隣接するノードに対して同様のことを繰り返して探索対象ノードを見つけることができる．また，根ノードから各ノードへの最短距離も求めることができる．

図 1 はノード 0 を根ノードとした際の Frontier を用いた幅優先探索の流れを表している．まず，Frontier にノード 0 と隣接しているノード 1，3，5 が格納される．このときに dist 配列

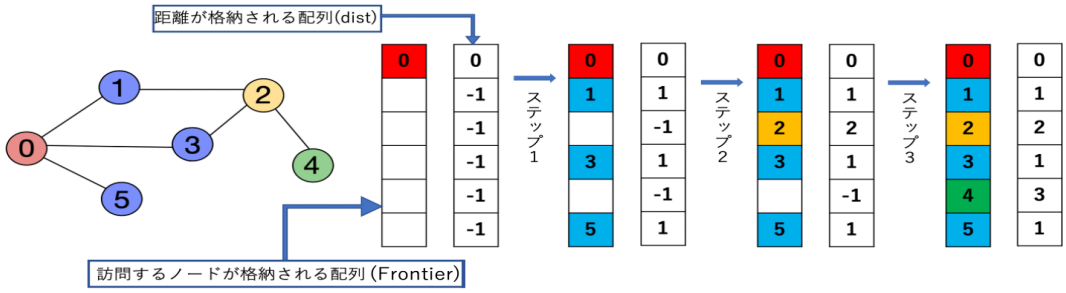

図 1　Frontier を用いた幅優先探索の流れ

の 1，3，5 番目にそれぞれ 1 が格納される．次に，ノード 1，ノード 3，ノード 5 に隣接するノードが Frontier に格納され，距離計算を行う．この動作をすべてのノードが Frontier に格納されるまで繰り返すことで検索対象ノードとそれまでの最短距離を得ることができる．

2.2　FPGA

Field Programmable Gate Array (FPGA) とは任意の回路をプログラミングによって繰り返し実装可能なハードウェアチップである．FPGA は実装された各回路の並列性を利用した，並列度の高い処理が可能である．

また，FPGA は Graphics Processing Unit (GPU) と比べて低レイテンシであり，動作周波数が低く，消費電力を抑えることができる．このような利点から，近年，FPGA は機械学習やビッグデータ処理など様々な分野で利用され始めている．

従来，FPGA は Verilog や VHDL のようなハードウェア記述言語によってプログラムされることが一般的であり，大規模なプログラムの開発には多大なコストを要していた．しかし，近年 C/C++や OpenCL 等の高水準言語を利用して FPGA のプログラムを可能にした，高位合成と呼ばれる技術が確立し，開発コストを抑えることが可能となっている．

2.3　CIRCUS

CIRCUS (Communication Integrated Reconfigurable CompUting System) [1] とは，筑波大学計算科学研究センターで開発されている，FPGA 間通信フレームワークの一つである．

近年の FPGA には，最大で $100Gbps \times 4$ の通信性能を持つ外部通信用光リンクを利用できる．この外部通信用光リンクは FPGA 上の回路と直接接続されているため，超低レイテンシで他の FPGA と通信することが可能となる．この特徴を活用する技術が FPGA 間通信フレームワークである．

CIRCUS は，複数の FPGA 間で巨大なパイプラインを構成するというパイプライン通信を実現することを可能にする．また，FPGA 内通信に利用される Channel を FPGA 間に拡張した CIRCUS Channel により，通信と演算が一体となったパイプラインを OpenCL で記述できる．

さらに，CIRCUS ではコードジェネレータが開発されており，CIRCUS に関するコードを通信定義ファイル (XML ファイル) から自動生成させる設計となっている．そのため，CIRCUS channel の数やデータ型などアプリケーションに依存して変化する部分でも，通信定義ファイルを変更するだけでよいという利点もある．

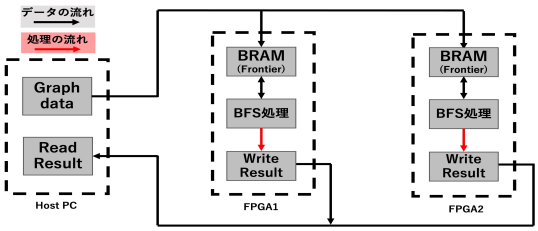

図 2　提案手法の全体の概要 (処理とデータの流れ)

CIRCUS の基礎通信性能の評価では，隣接する FPGA 間通信で最小レイテンシ $0.5\mu s$，最大 90.2Gbps を達成している [1]．さらに，比較手法 SMI [2] との性能比較において 3 倍高いスループットと，同程度の通信レイテンシを達成している．

3　関 連 研 究

3.1　FPGA を用いた高速化の事例

FPGA を利用した処理の高速化に関する研究は，様々な分野で行われている．ゲノム科学の分野では，Varma らは参照ゲノムに対して何百万もの短い文字列（リードと呼ばれる）をマッチングさせるショートリードアセンブラの一部を FPGA で実装した [3]．また，グラフ分析の分野では，Zhou らは PageRank [4] の高速化手法を提案した [5]．

3.2　幅優先探索に関する研究

幅優先探索の高速化には、分散処理やハードウェアアクセラレータである GPU や FPGA が利用されている．幅優先探索の分散処理は Graph500 ベンチマーク [6] で採用されており、スーパーコンピュータで様々な高速化が検討されている。

3.2.1　複数 FPGA を利用した幅優先探索の高速化の事例

複数の FPGA を利用した幅優先探索の高速化として Dai ら [7] の研究がある．Dai らは，マルチ FPGA アーキテクチャに基づく大規模グラフ処理システム ForeGraph を提案した．ForeGraph では，各 FPGA がグラフ全体のパーティションをオフチップメモリに保存するだけであるため，各 FPGA は競合なしにグラフ処理を並列に実行することが可能となる．その結果として，従来の FPGA ベースのシステムと比較して 5.89 倍の高速化と，2.03 倍の平均スループット向上を達成している．

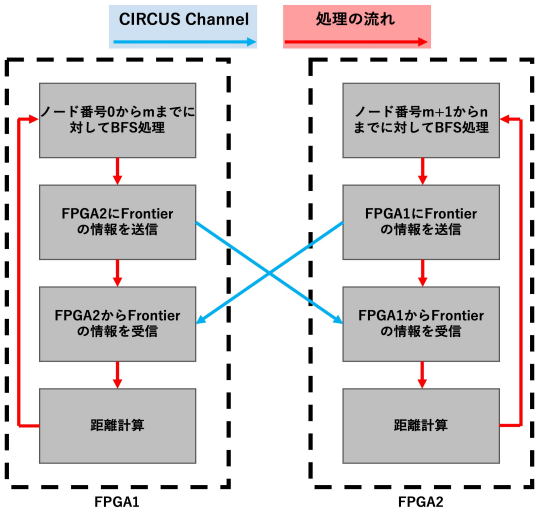

図 3　提案手法による幅優先探索の実装の詳細

4　提 案 手 法

本研究では Frontier を用いた幅優先探索を 2 つの FPGA を用いて実装するための手法を提案する．また，グラフの表現方法として，CCS 方式を採用する．CCS 方式では，グラフデータを 2 つの配列を利用して表すこととし，その配列を $ptrs$ と $elements$ と記載する．$ptrs$ は，$ptrs[i]$ と $ptrs[i+1]$ の差がノード i の持つ隣接ノード数を表す配列であり，$elements$ は，$elements[ptrs[i]]$ から $elements[ptrs[i+1]]$ までに，ノード i の隣接ノードのノード番号が格納される配列である．

4.1　提案手法の処理の概要

全体の提案手法の概要を図 2 に示す．全体の処理は Host 側で行われる処理と 2 つの FPGA 側で行う処理に分かれる．まず，幅優先探索の対象グラフのデータを Host 側で受け取り，$ptrs$ と $elements$ を二分割する．これはそれぞれの FPGA へ異なる情報を持った配列を送信するためである．分割は $elements$ がおおよそ半分になるようにする．これは，単に $ptrs$ を前半と後半で分割してしまうと，片方の FPGA に $elements$ の情報が偏ってしまう可能性があり，FPGA の処理が一方に偏り，性能のボトルネックとなる可能性があるためである．

分割したグラフデータと Frontier は，FPGA チップ上の BRAM に配置される．BRAM は FPGA チップ上の回路が 1 サイクル以内にアクセス可能であり，非常に高速なメモリである．しかし，その容量は小さく，数百 KB〜数百 MB 程度であることが多い．

各 FPGA へグラフデータの送信が終わったあとで，各 FPGA は幅優先探索を行う．幅優先探索が終わった後，根ノードからの各ノードへの最短距離が各 FPGA から Host へ送信される．最後に，FPGA から得られた最短距離と，CPU で幅優先探索を行い得られた各ノードへの最短距離を比較する．

4.2　幅優先探索の実装

幅優先探索にはトップダウン・アプローチ，ボトムアップ・アプローチ，トップダウン・アプローチとボトムアップ・アプローチを状況によって切り替えるハイブリッド型の実装方法がある．本研究では，その中でもトップダウン・アプローチを採用した幅優先探索を FPGA 上に実装する．

トップダウン・アプローチは，Frontier に含まれるノードに対して隣接ノードを探索し，発見された隣接ノードが Frontier に含まれる場合は操作は行わず，含まれていない場合は，その隣接ノードを Frontier に追加する．Frontier に属するノードに対して，すべての隣接ノードが訪問済みかどうかの確認を行うため，冗長な計算が発生してしまう．

2 つの FPGA で幅優先探索を実装する詳細を図 3 に示す．各 FPGA に送信されたグラフデータにおいて，$ptrs$ が 0 番目から m 番目と $m+1$ 番目から n 番目の場合，FPGA1 はノード番号が 0 番目から m 番目のノードのみに対して探索を行い，FPGA2 はノード番号が $m+1$ 番目から n 番目のノードのみに対して探索を行う．つまり，各 FPGA は Frontier に含まれるノードが自身が探索すべきノードかどうかを判断したあとで，そのノードが自身が探索するべきノードであったときにそのノードの隣接ノードを探索する．

各ステップが終了したあとで Frontier を更新する．各 FPGA は自身が探索するべきノードのみに対して探索を行っているため，Frontier を更新するときに各 FPGA の Frontier 情報を同期させる必要がある．このとき FPGA 間高速通信システム CIRCUS を利用する．各 FPGA は自身の探索対象のノードが新たに Frontier に追加されたとき，そのノードの情報を CIRCUS Channel を利用してもう一方の FPGA に送信する．通常，このような同期は性能のボトルネックとなるが，CIRCUS を利用することで超低レイテンシに同期を行うことが可能となる．

また，本研究では根ノードから各ノードへの距離計算は幅優先探索を行う処理と分けて実装を行った．ノード i がステップ t で訪問された場合，根ノードからノード i への距離は t となる．また，すでに訪問済みのノードが Frontier に再度追加されることはない．そのため，ステップ u でノード k が訪問され，かつノード k が Frontier に属していない場合に根ノードからノード k までの最短距離を u とすればよい．したがって，この処理で距離計算を行うことで，各 FPGA の幅優先探索の処理と距離計算を切り離す事が可能となる．

5　実　　験

5.1　実験設定とデータセット

本研究では，データセットとして test データ以外は Network Data Repository [8] のグラフを用いた．本実験で使用した実験環境を表 1 に，データセットの詳細を表 2 に示す．

実験では，データセットを事前に CCS 方式に変換しており，その変換されたグラフデータを使用する．また，図に示す実行時間はランダムに選ばれた根ノードに対して，幅優先探索を 10 回行ったときの平均実行時間とする．

表 1　実 験 環 境

OS	Linux version 3.10.0
CPU	Intel(R) Xeon(R) CPU E5-2660 v4 × 2
Boad	Bittware 520N
FPGA	Intel Stratix 10 GX 2800
ロジックエレメント	2800000
BRAM	229MB
MLAB	15MBits

表 2　データセットのノード数とエッジ数

データセット名	|V |	|E |
insecta-ant-colony1-day11	113	3454
test	320	1938
bio-SC-TS	636	3959
reptilia-tortoise-network-fi	787	1197

5.2 実 験 結 果

複数 FPGA による幅優先探索を正しく実装できているかを確認するために，CPU を利用した場合の根ノードから各ノードへの最短距離と，提案手法で得られた最短距離を比較した．その結果，各データセットについて提案手法が正しく動作していることを確認できた．

表 2 で示した各データセットに対して，提案手法を実行した際の処理時間と，単一の FPGA を用いた場合の幅優先探索の処理時間を図 4 に示す．単一の FPGA を用いた手法では，表 1 と同じ FPGA を利用した．

図 4 より，提案手法のほうが処理時間が長くなっていることがわかる．CIRCUS には，Virtual Channel が未実装であるため通信網に循環があるとでデッドロックが発生しうる．提案手法では，探索が一度終了した後で Frontier の同期のために，CIRCUS Channel の書き込み，読み込みを 2 つのカーネル間で循環させている．そのため，提案手法ではデッドロックが起こりうる．実際には，一方の FPGA で CIRCUS Channel への書き込みを行っている場合，もう一方の FPGA では CIRCUS Channel からの読み込み段階で処理が一時停止してしまう．この処理の停止が，提案手法のほうが処理時間が長くなってしまった原因であると考えられる．

また，本手法では大規模なグラフデータを処理対象とした場合処理を行うことができなかった．これは CIRCUS Channel の読み込み量，書き込み量共に大量になってしまい，処理の停止が非常に長くなってしまったことが原因と考えられる．

6 ま と め

本研究では FPGA 間通信フレームワーク CIRCUS を利用した複数 FPGA によるグラフ幅優先探索手法を提案した．CIRCUS を利用することで FPGA 間通信を超低レイテンシに行うこと，複数 FPGA を用いることで各 FPGA の探索コストを削減することを組み合わせ，効率的な幅優先探索を提案した．

結果として，小規模なグラフデータに対しては，複数 FPGA を用いた幅優先探索を正しく実装できたことを確認できた．

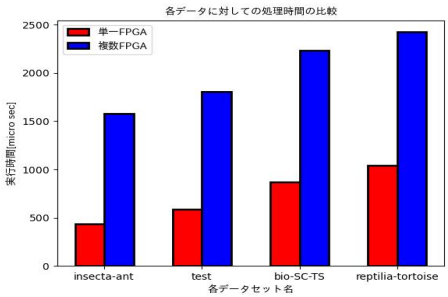

図 4　処理時間の比較

しかし，提案手法は単一の FPGA と比べ処理時間が大幅に伸びてしまった．また，対象グラフが大規模な場合，デッドロックが発生してしまい，処理を行うことができなかった．

今後の展望として，大規模なグラフに対応するためにデッドロックを解消するアルゴリズムの提案を進めていく予定である．

謝　　辞

本研究は、文部科学省「次世代高性能計算機基盤・応用研究開発プログラム」（次世代スーパーコンピュータ通信基盤の開発）の一環として実施されました．本研究の一部は国立研究開発法人新エネルギー・産業技術総合開発機構 (NEDO) の委託業務 (JPNP20006)，筑波大学 CCS 多地点連携研究制度，日本学術振興会科研費番号 JP22H03694，JP22K17895，JP19H04114，JST CREST 課題番号 JP-MJCR22M2，AMED 課題番号 JP21zf0127005 によって支援されています．

文　　献

[1] Norihisa Fujita et al. Performance evaluation of pipelined communication combined with computation in opencl programming on fpga. In *2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pp. 450–459. IEEE, 2020.

[2] Tiziano De Matteis et al. Streaming message interface: High-performance distributed memory programming on reconfigurable hardware. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–33, 2019.

[3] B Varma, Kolin Paul, and M Balakrishnan. Fpga-based acceleration of de novo genome assembly. In *Architecture Exploration of FPGA Based Accelerators for BioInformatics Applications*, pp. 55–79. Springer, 2016.

[4] Lawrence Page et al. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[5] Shijie Zhou et al. Optimizing memory performance for fpga implementation of pagerank. In *2015 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, pp. 1–6. IEEE, 2015.

[6] Richard C Murphy et al. Introducing the graph 500. *Cray Users Group (CUG)*, Vol. 19, pp. 45–74, 2010.

[7] Guohao Dai et al. Foregraph: Exploring large-scale graph processing on multi-fpga architecture. In *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pp. 217–226, 2017.

[8] Ryan A. Rossi and Nesreen K. Ahmed. The network data repository with interactive graph analytics and visualization. In *AAAI*, 2015.