Direct Universal Access: Making Data Center Resources Available to FPGA

Ran Shu[1], Peng Cheng[1], Guo Chen,2[1], Zhiyuan Guo,3[1], Lei Qu[1], Yongqiang Xiong[1], Derek Chiou[4], Thomas Moscibroda[4]

[1]Microsoft Research, [2]Hunan University, [3]Beihang University, [4]Microsoft Azure

動機

- FPGA の大規模利用がちょくちょく話題に上がる Microsoft さんの論文。
- 実際に使われているかはともかく、何を問題視していて、どのように解決したかという過程が知りたい。

Contents

概要

概要

- FPGA を大規模データセンターにデプロイすることを考える。
- このとき、FPGA からデータセンターのリソースをうまく利用するのは ちょっとむずかしい。
  - オンボードのホストのリソースに関してはまぁ良い。
  - 問題はデータセンター内のリモートホストのリソースを利用することを考えたとき。
- FPGA からデータセンター内リソースへのアクセスを抽象化して同一方法で様々なリソースを取り扱えるようにした！（Direct Unversal Access (DUA)）
- 評価したところ実装に必要な回路サイズも小さく、性能も良かった。

背景と目的

背景

- 大規模 FPGA アプリケーションを構築するには複数の FPGA でコミュニケーションとったり、それ以外のリソースともやり取りできてほしいが、概ね以下のような問題がある。
  1. 別ホストに対して、様々なリソースを、様々な通信方式で要求することはプログラムを非常に複雑にする。
  2. server-centric である。FPGA → SSD(Remote) は、FPGA → CPU(FPGA Host) → CPU(Remote Host) → SSD (Remote Host)
  3. リソースマルチプレクシングができない。これは開発者が処理を記述する必要がある。同じリソースを使用する複数 FPGA が存在する場合、同時アクセスなど問題はより深刻化する。

DUA Overview

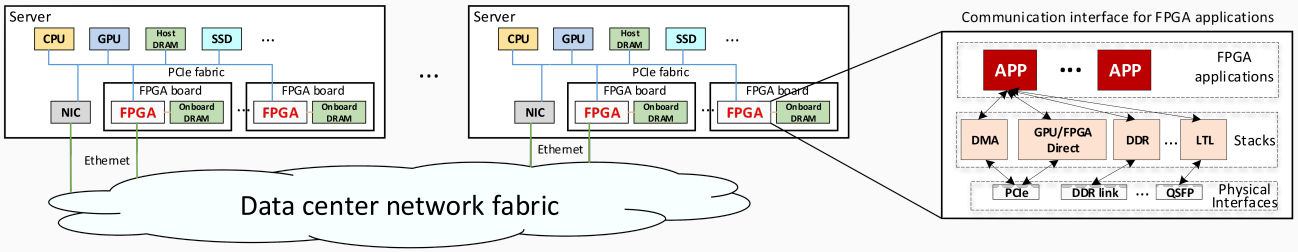
Communication Architectureが固定的

・例えば、ToR越しに直接NICで接続された2台のFPGAの環境で、同じPCIeドメインにいる場合のみリモートで利用が可能
  ・ただし帯域を食いつぶすことができず無駄になる
  ・スケーラビリティーにも難

リソースへのアクセス性とunfied naming

・DCリソースという面でいえば潜在的にFPGAから利用できるリソース（computation/memory resource）は大規模である。
・しかし実際に利用を考えるとFPGAがリモートサーバのDRAM、SSD、GPUなどのリソースを利用するのは難しい。
  ・これはnamespaceがlocal serverであることに起因している。

・unified naming schemeを定義し、リモートから特定のリソースにアクセス／利用できるようにしたい。



Resource Multiplexerが貧弱

・DCリソースをプールとして見るときにFPGAで考えるべきはマルチプレクサである。
  ・ex. 同一リソースへの複数アクセス時のmux/demux
  ・実装に354行程度HDLコードが追加で必要

・現時点で、一般的な物理インターフェスのmulxiprexing schemeは存在しない。（先行研究はある）
  ・同じPCIeバスを利用する別リソースへの同時アクセスも難しい
  ・ローカルホストDRAMと、SSDへの通信など

・更に異なるインターフェース間、Stack間のmultiplexingは現在考えられていない。

通信インターフェースの実装が困難

・アプリケーション開発者に高いプログラミングスキルが求められる。
・特にベンダー依存、実装依存が強い場所でもある。
  ・基本的に既存のStackは利用できる。
  ・ただしベンダー同士で相互互換性がなかったりなどもする。
・異なる通信インターフェースにかかる実装コスト一覧

Resource	Communication stack	LoC
Host DRAM	DMA	294
Host CPU	FPGA host stack	205
Onboard DRAM	DDR	517
Remote FPGA	LTL	1356

非効率的なCommunication Stack

・コミュニケーションStackの実装・選択が効率に大きく影響を与える。
  ・例えば、PCIeを通したFPGA-to-FPGAの通信メカニズムでは通信開始前の同期処理が入る。
  ・例えば、リモートホストのDRAMへアクセスする際は、LTLを利用しないとCPUを中継する必要が発生し性能が下がる。

問題の解析と理想のアーキテクチャ

問題 1

- hoghoge

問題 2

- hoghoge

問題 3

- hoghoge

Desicred Communication Architecture

- hogehoge
- fugafuga

目的

目的

- 目的：FPGA の世界に理想的なコミュニケーションアーキテクチャ（DUA）を導入する。

1. 共通のコミュニケーションインターフェースの導入
2. 全てのリソースに対してアドレスを付加し、リソースの場所に関わらずアクセス可能にする。
3. ルーティングやリソースマルチプレクシングできるようにネットワークサービスを導入。

DUA overview / DUA components

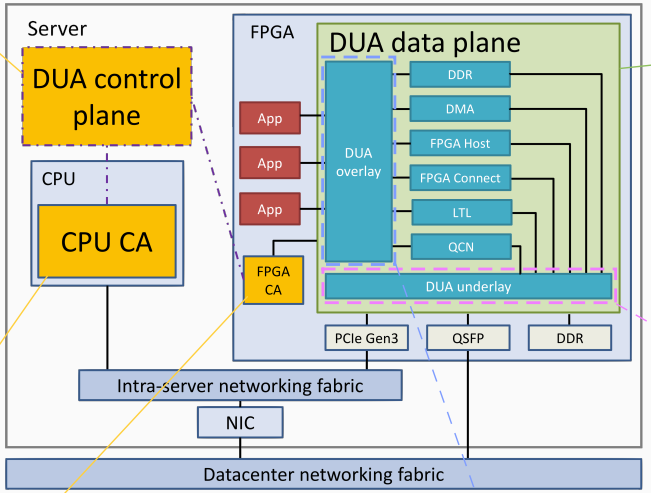
DUA Overview

DUA : Communication Architecture （overlay network）

Control plane
・CPU / FPGA Control Agent
・以下のmanagementを行う
　・全てのリソース情報
　・ルーティング
　・コネクション
・CPU CA
　・ロジカルな処理はここで行う
・FPGA CA
　・local resourceのモニタリング
　・データプレーンへの制御情報の配信

CPU CA
・全てのリソースにUIDを付与する
　・全てのFPGA CAからリソース情報を収集
　・ホストメモリやGPUの情報なども収集
・ルーティングの管理をする
　・Interconnection tableを管理
　・これによりルーティングを制御
・コネクションマネジメント
　・セキュリティの担保もする。
　・ローカル／リモートリソースの接続管理
　・コネクション接続／切断をAppに伝える

FPGA CA
・FPGAボード上のリソースをモニタリング
　・オンボードDRAM、FPGA Applicationなど
・FPGA CAはそのホストのCPU CAに対して、上記の情報を通知する。

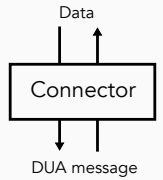


Data plane
・HWで実装
・low-cost設計
・主に3種のcomponents
　・DUA overlay
　・Stack
　・DUA underlay

DUA underlay
・Hard IPとリソース共有の管理
・外部攻撃からDUA Stackを保護
・データ送出失敗の制御

・Policyベースで管理
・仮想トランザクション層

Connector
・データ変換



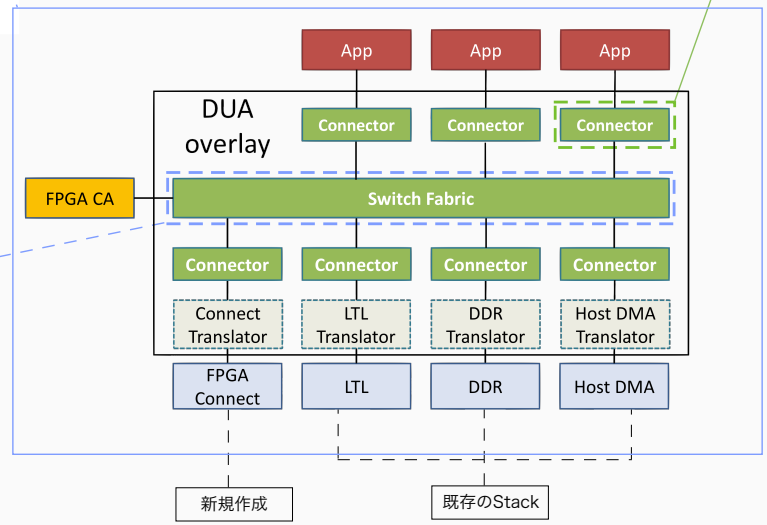
・maintain/lookup forwarding table
　・各connectorは自身のtableを持つ

・Access Controlが可能

・Transport protocolの再実装なし

Switch Fabric
・インターネットルータと同じ役割
・着信コネクタ → 送信先コネクタ

・現在の実装では以下のStackに対応している。
　・FPGA Connect
　　・これは独自実装のStack
　　・FPGA間通信
　・LTL
　・DMA
　・DDR
　・FPGA Host



DUA Communication Interface

- hogehoge

DUA Control Plane

- hogehoge

DUA Data Plane

- hogehoge

Evaluation

Evaluation

- hogehoge

Microbenchmark

- hogehoge

Application Benchmark

- hogehoge