Vol.2021-OS-151 No.1
2021/3/1

MEC FPGA クラスタにおける TCP/IP の実現

並木 美太郎[1]、天野 英晴[2]

概要：筆者らは、FPGA ボードを高速リンクで接続した MEC(Multi-access Edge Computing)システムを研究している、FPGA ボードには、Xilinx 社の Zynq が搭載され、ARM と FPGA で処理を行える。開発環境は PYNQ を用い C ライクな、高位合成言語と Python で処理を行える。これら FPGA ボードを高速リンクで結んだ FPGA クラスタで､OS を直すことなく､柔軟にユーザランドでハードウェア処理を 行いたい｡そこで､FPGA ボード間通信で tun/tap 仮想デバイスを用いて、TCP/IP を実現し、柔軟な通信路設計や FPGA 処理の仮想化を行った

キーワード：FPGA　MEC　　Pynq　Zynq　　tun/tap デバイス TCP/IP

An Implemwntation of TCP/IP for MEC FPGA Cluster

Mitaro Namiki[†1]　Hedeharu Amano[†2]

Keywords：FPGA　MEC　　Pynq　　Zynq tu/n/tapdevice　　TCP/IP

1．はじめに

Multi-access Edge Computing(MEC)は、第 5 世代移動通信(5G)の基地局に設置された計算システムにより、複数のエッジ端末からの要求に応じて処理を行う方式である。

5G でサポートされている低遅延･高帯域通信により､A.I. 技術を利用したスマートシティの交通管理や、高度な工場制御などタイミングクリティカルな処理を MEC で行うことが期待されている。

小型デバイスを用いたエッジコンピューティングとは異なり、より強力な計算能力を必要とする画像認識や音声認識を MEC で行うことができる。

また、クラウドとの通信のための遅延時間が大きく不確実性が高いという問題も回避できる。

しかし､データセンターのサーバと比較して､MEC のサーバは、建物の上に設置される基地局に設置しなければならないため、電力やコストが厳しく制限される。

FPGA コンピューティングはコストと消費電力に優れた MEC サーバとして注目されている。

また、エッジからの要求を直接 I/O で受け付け、ハードワイヤードロジックにより決まった時間で処理することで、タイミングクリティカルなジョブに対応できることも FPGA コンピューティングの重要な利点の一つである。

特に、基地局への要求が多い場合には、マルチ FPGA システムが有利である。

ここでは､MEC のコアコンピューティングサーバとして、スタンドアロン型の PYNQ クラスタを提案する。

このクラスタは、M-KUBOS[1]と呼ばれるハイエンドの Zynq ボードで構成されており､低コストかつ高性能な GTH シリアルリンクでボード間が接続されている。

従来の Zynq クラスタである Zedwulf[2]は、相互接続に PS(CPU)部の Ethernet を使用して相互結合を行っていたが M-KUBOS では PL(FPGA)部を高性能リンクで直結している。

PYNQ の Ubuntu Linux は Zynq PS 部のクアッドコア A53 上で動作するため、ジョブの分散や PL 部のコンフィグレーション、複数ボードの入出力を管理することができ、電力効率の悪い x86 サーバは不要になる。

タイミングクリティカルなジョブは PL 部ハードウェア上で実行し、深層学習の実行には PL を使うだけでなく、エッジ DSA(Domain Specific Architectures)[3]などを PS 部に接続し利用できる。

さらに M-KUBOS よりも低価格な FPGA で構成された拡張クラスタ(Flow-in-Cloud、FiC)を接続することも可能である。

1 東京農工大：Tokyo University of Agriculture and Technology
2 慶応大学：Keio University

本稿ではこの PYNQ クラスタ[4]の構成を述べ、PYNQ を用いてクラスタを構築するために必要な機能を備えた管理システムを実装し、コンフィグレーションにかかる時間を計測、評価を行った。

2．MEC

Multi-access Edge Computing(MEC)は、5G モバイルネットワークの標準規格として ETSI(European Telecommunications Standards Institute)で標準化が進められている。
これによると、1 つ以上のタイプのアクセス技術を含むアクセスネットワークのエッジで、利用者に近接して IT サービス環境とクラウドコンピューティング機能を提供するものである。
5G 無線技術の進歩を背景にしているが、5G に限定されず、WiFi、LPWA、有線ネットワーク環境でも同様の概念が成立する。
MEC のメリットをまとめると以下のようになる。
(1)エッジデバイスは、近くの MEC にタスクをオフロードできる。
(2)アプリケーションをローカル環境で動作させることができ、応答時間やユーザの利便性が向上する。
(3)クラウドとのネットワーク帯域を節約し、ネットワークの輻輳を軽減することができる。
この特徴を利用し、工場制御、スマートシティのトラフィック管理、セキュリティ制御など、様々な用途が期待されている。

この MEC サーバには以下の要件がある。
(1)建物の上に設置される基地局に設置するため、低消費電力・低コストであること。
(2) タイミングクリティカルジョブを実行するため、スケジューラビリティが高く、将来性があること。
(3)エッジデバイスからの直接要求にも対応できる強力で多様な I/O 能力を持つこと。

3．M-KUBOS ボード

3．1 ボードの構造

PALTEK の M-KUBOS ボードは[1]、MEC のキーコンポーネントとして筆者らのコンセプトに基づいて製品化された。このボードは、最上位の Xilinx UltraScale+ xczu19eg と 2 枚の DDR4 DRAM モジュールで構成されており、それぞれが PS(Processing System)と PL(Programmable Logic)に接続されている。
図 1 に示すように、PS には各種インタフェースが、PL には高速シリアルリンク GTH、GTY が多数装備されている。PS は ARM Cortex A53 クアッドコアと Cortex R5 クアッドコアを搭載しており、PL は 1143K のロジックセル、70.6Mb の Block/Ultra RAM、1968 個の DSP を搭載した Ultrascale+ FPGA である。
PL の規模はハイエンド FPGA と比較しても遜色ない。
両者はパッケージ内でデュアル AXI バスにより緊密に接続されている。

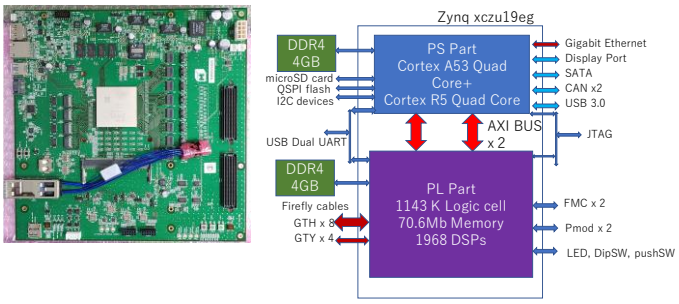

図 1　M-KUBOS Zynq Ultrascale+ボード

3．2　スタティック領域

図 2 に、M-KUBOS 上の Zynq Ultrascale+チップ内の内部回路を示す。

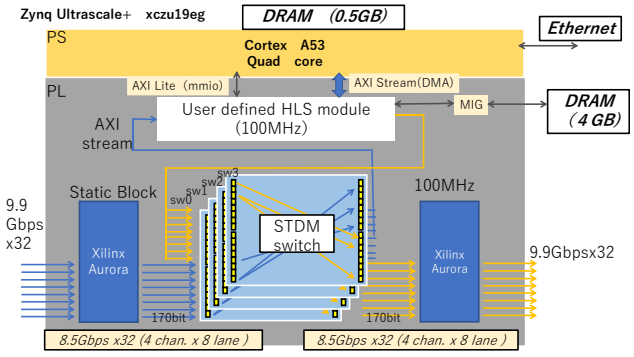

図 2 Zynq UltraScale+の内部回路

PL はスタティック領域とユーザ HLS 領域の 2 つの領域に分かれている。
スタティック領域には、GTH ポートを介して他のボードと高速シリアルデータを入出力するための Xilinx の Aurora IP が搭載されている。
現在、実装を容易にするため、他のボードとの接続には、9.9Gbps の転送レートで Firefly ケーブルを使用している。
シリアルデータは 2 クロックあたり 170 ビットのパラレルデータに変換され、スタティック時分割多重(STDM)スイッチで通信を切り替えている。
STDM はパケットスイッチングに比べて軽負荷のトラフィックには効率が悪いが、仮想的な回路を形成し、これに対して一定のタイミングスロットが割り当てられるため、バンド幅が保証され、レイテンシが予測可能になる。
これらの特徴により、STDM は MEC のタイミングクリ

ティカルな通信に適している。

現在の実装では、全体の転送帯域は 272Gbps となり、ボード当たりの遅延は約 550ns となっている。

FireFly ケーブルは 4 本のシリアルリンクを束ねているため、独立した STDM スイッチが 4 つ用意されている。

これら 4 つのスイッチの接続ボードは同じであるが、独立したスロット数のスイッチを非同期的に動作させることができる。

これらのスイッチは、リンクアグリゲーションによるバンド幅の増強や、スロット分配によるスロット数の削減に用いることができる[3]。

３．３　HLS 領域

HLS 領域は、C/C++で記述され、Vivado HLS で合成されたユーザー定義アクセラレータ用の領域である。

4 つの STDM スイッチは、AXI Stream インタフェースを介して HLS モジュールに接続されている。

したがって、HLS モジュールは、接続された他のボード上の HLS モジュールとの間で、4 つの 170 ビットの AXI Stream 入出力を持たせることができる。

図 3 に Zynq Ultrascale+の Vivado Block Design の大まかな構成を示す。

各モジュールは Vivado Block Design のブロックに対応している。

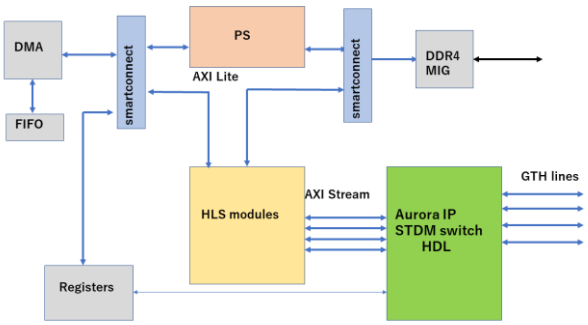

図 3　ZynqUltraScale+の Block Design

PS とのインタフェースとして、AXI-lite バスが 2 本用意されており、HLS モジュールは AXI-lite バスによる共有メモリを介してデータを取得し、結果を返すことができる。また、起動・停止制御も PS ソフトウェアから行うことができる。

高性能な通信のために、DMA コントローラと FIFO が用意されており、AXI Stream インタフェースで接続することができる。PL に搭載されている 4GB の DDR4 DRAM は、MIG DRAM コントローラを介して AXI Master インタフェースでアクセスすることができる。

すべてのモジュールは Block Design の IP の形で用意されており、ユーザーは必要なモジュールを選択し、設計した HLS モジュールと接続することで、DRAM や他のボードとの接続を含めた設計を行うことができる。

４．M-KUBOS/PYNQ クラスタ

４．１　PYNQ の導入

PYNQ(Python productivity for Zynq)[6]は、元々は Xilinx 社が PYNQ-Z1 用に開発したオープンソース・ソフトウェア・プラットフォームである。PYNQ-Z1 は、比較的小型の Zynq チップを搭載した FPGA ボードである。

PYNQ 環境上では Python や Jupyter Notebook が動作し、各種ライブラリを利用して AI アプリケーションを簡単に実装できることから、AI プラットフォームとして注目された。

Zynq は組み込みシステムでの利用を想定しており、PS の ARM 用のソフトウェアは PS のハードウェア開発と合わせて Vivado と連携したツールにより設計する。これにより生成したマシンコードとコンフィグレーションデータをまとめて Zynq ボードにダウンロードして実行する。

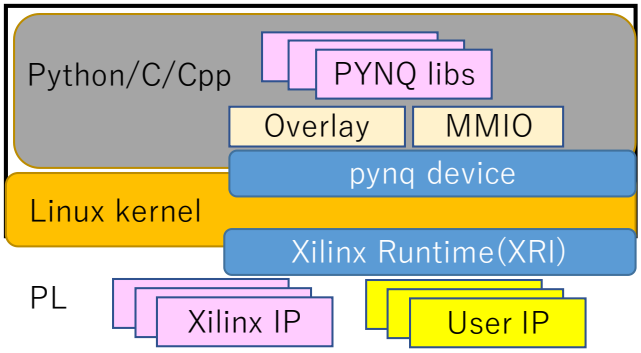

図 4Pynq の構造

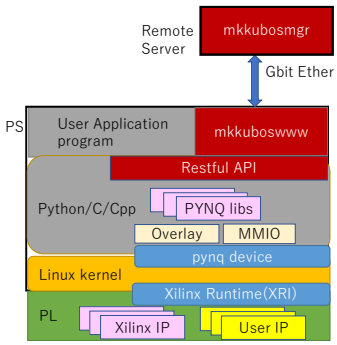

図 5 ノードの構成

この使い方は、PS と PL で実行される多数のタスクを Zynq クラスタに受け入れて分散実行しなければならない MEC には不向きである。

つまり、MEC のための Zynq クラスタの使い方は、組み込みコンピューティングというよりは、クラスタの FPGA に近いものになる。PYNQ は組み込み小型ボード向けに開発され

たが、Zynq で従来サポートされてきた Peta-Linux とは異なり、Ubuntu Linux 上に構築されており、Linux サーバ用のソフトウェアスタックを構築することができる。また、PL を制御するためのレイヤがオーバーレイとして統合されているため、PS を停止することなく PL をコンフィグレーションしたり、Python や Jupyter Notebook、C/C++を含む他のプログラミング言語とのインタフェースをとることができる。

このように、PYNQ を M-KUBOS の各ボードに導入することで、クラスタ構築の基盤として用いることができる。
しかし、PYNQ は元は 1 ボード向けに開発されたが、M-KUBOS は MEC として用いるため、複数のボードを群管理する必要がある。
そのため、これをサーバとして動作させるためには、後述の拡張機能が必要である。
FiC においてもこのような群管理システム[4,5]を運用し、アプリケーションの開発に役立てている。

４．２P　YNQ クラスタの現状

現在の PYNQ クラスタは、図６に実機の写真を示す。4 枚の M-KUBOS ボードを接続して構成されている。
現在は 5 入力 5 出力のスイッチを使用しているが、クラスタのサイズが大きくなれば、簡単に 9 入力 9 出力まで拡張することができる。

図６　4 枚の M-KUBOS ボードによる Pynq FPGA クラスタ

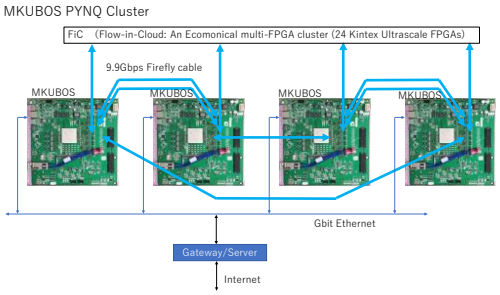

図 7 pynq クラスタの概念図

先に述べた通り、Firefly ケーブルの性質上、現在のクラスタには 4 つの重複したリンク/スイッチが用意されているが、図の混雑を避けるために省略している。

ポート 0 は AXI Stream インタフェースで HLS モジュールに接続し、ポート 1 に FiC を接続してさらに大きなクラスタにすることも可能である。
(1) 他の 2 つのポートを用いて M-KUBOS ボード間にリングネットワークを構築する。
(2) 重複したリンクは、リンクアグリゲーションに利用してバンド幅を増やすことができる。

５．Pynq 利用の課題

システム構築者が作成した FPGA の PL 部のハードウェアは、OS を修正することなく、ユーザプロセスの pynq ライブラリで利用できる。
ユーザレベルで拡張したハードウェアを処理できる。

したがって、pynq クラスタ上のハードウェアリソースを、GTH の FPGA 間リンクもユーザプロセスレベルで適切に扱えることが課題となる。

また、FPGA 部で TCP/IP などの IP パケットのデータを扱うには、通常、NIC ドライバを実装する必要があるが、煩雑である。
そこで、Linux の tun/tap デバイスを用いて、GTH リンクに IP を実現する。

６．tun/tap による IPover GTH の実現

６．１ tun/tap デバイス

Tun/tap デバイスは Linux、FreeBSD の OS カーネルで提供される仮想ネットワークデバイスである。VPN の実現で使われる。

eth0 のような仮想 NIC と/dev/net/tun のようなキャラクタデバイスが対になる。仮想 NIC 側は IP アドレスを割り当てることができ、通常のネットワーク操作が可能である。tun は、L3 デバイス、tap は L2 デバイスである。
Tun/tap に送受信されたパケットは、/dev/net/tun のキャラクタデバイスに tun なら IP パケットが、tap ならイーサパケットが入出力される。VPN では、インタネット上にソケットにより、パケットをトンネリングする。ソケットで送られて来たパケットをｋｙラクタデバイスに write することで、仮想 NIC からパケットを受信できる。
逆に仮想 NIC に送ったパケットは、キャラクタデバイスで read できるので、read してインタネット上のマシンに転送することで VPN が実現される。
キャラクタデバイスには IP パケットが現れる。VPN ではインタネットとソケットでマシン間でパケットを中継する。マシン間のデータ転送はどのような通信路でも良い。

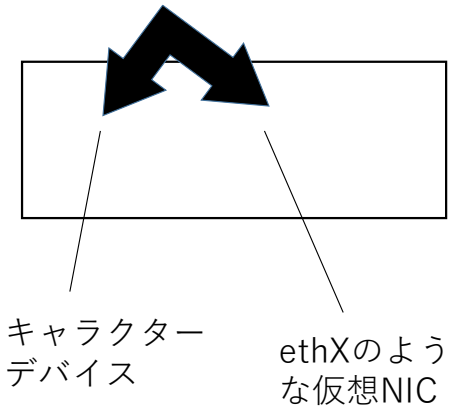

図８./tap デバイスの概念図

NIC のドライバがなく、バイトストリームの入出力があれば、IP パケットをその通信で送受信できる。

６．２tun/tap デバイスによる IP overGTH

本研究では、tun/tap デバイスを用いて GTH リンクに IP パケットを送受信する。図 9 に構成図を示す。FPGA で実装された GTH の読み書きは C プログラムで行えるので、GTH で受信した IP パケットを/dev/net/tun キャラクタデバイスに write することで、tun デバイスで IP パケットが処理される。

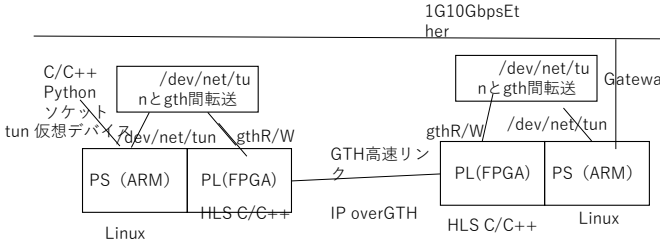

図９．tun/tap による IPover GTH

/dev/net/tun キャラクタデバイスから IP パケットを read して、GTH に write すれば、FPGA 間の GTH リンクに IP パケットを送信できる。キャラクタデバイスと GTH 間の転送プログラムはマルチプロセス、select/epoll などを用いて、GTH、TUN/TAP 間転送を実現した。TCP/IP、UDP/IP、ICMP の動作を確認した。tun デバイスを用いて、通常 EtherNIC に経路設定することで、また M-KUBOS を GTH-インタネットゲートウェイにできた。ゲートウェイに限らず GTH のリンクを通常の TCP/IP で扱える。同時に、 MEC で FPGA にオフロードした処理を TCP/IP で呼びだせる機能を実現した。

７．tun/tap による IPover GTH の UDP エコー処理による評価

提案方式を用いて UDP によるエコー処理の評価を行った。プログラムは通常のソケットを用いて作成できた。図 10 に処理の流れを示すクライアントは単に数 KB のデータをサーバの IP アドレスとポートを指定して UDP で送信しサーバは受け取ったパケットを単にクライアントへ返す。このデータとパケットは FPGA の PL 部を経由して、aGTH リンクへ転送されている。言語 C だけでなく、Python でも動作を確認しているこれらの処理は Zynq の PS 部の ARM で処理されたが、今回は実装しなかったが、このクライアントやサーバのエコー処理は FPGA で実装して、FPGA で IP パケットを処理できる。本研究は、通常の TCP/IP のインタネットと FPGA 上の IP パケットの橋渡しも可能である。

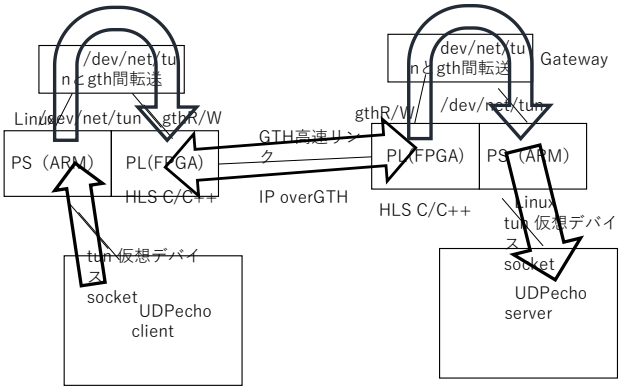

図 10.UDP エコー処理の流れ

本方式を用いて M-KUBOS ノード間で UDP、TCP で 10MB データをエコーバックした結果を表１に示す。同時に、KUBOS の Zynq が持つ 1Gether、同じ ARM の NVidea の Jetson AGX と Xeon に Intel 10GbNIC を用いたときの結果を表１に示す。性能最良値は Mbps である。TCP,UDP は 10MB データの実行時間である。TCP,UDP はマシン間では差はなく同じプログラムである。FPGA の GTH のプログラムが通常のソケットになったことでプログラミングが容易になった。

表 1. tun による IP over GTH の性能

(ms)	Mkubos1<-GTH->Mkubo2		x86Xon<->JetsonAGX	
通信路	1G	GTH	1G	10G
ICMP	0.159	0.205	0.277	0.203
TCP	1940	4837	2436	1349
UDP	789	2152	2528	1377
性能最良値	25.35	9.29	8.21	14.83
		10MB転送時間(ms)		

性能は GTH の 9.9Gbps、1 G,10G イーサのワイヤスピードにはなっ

ていない。しかし、FPGA を高位合成で利用し、従来の TCP/IP で使えることが特徴である。今後は端点に FPGA を用いて確認すること、本実装は MMIO による CPU 転送だったので、今後は DMA 転送などの高速化を行いたい。

8. 終わりに

本稿では FPGA クラスタで TCP/IP を実現するために tun/tap 仮想デバイスを用いる方式を提案し、性能を評価した。今後の課題は DMA 転送を用いるなど、性能向上である。

謝辞

本研究は JST CREST Society 5.0 を支える革新的コンピューティング技術「MEC 用マルチノード統合システムの開発」(JPMJCR19K1)の支援を受けたものです。関係者の皆さまに深く感謝の意を申し上げます。

参考文献

[1] PALTEK,” FPGA Computing platform M-KUBOS”,https;//www.paltek.co.jp/design/original /M-KUBOS/ accessed 020-12-21

[2] P.Moorthy, nd N.Kapre “Zedwulf: Power-Performance Tradeoffs of a 32-Node Zynq SoC Cluster”.2015 IEEE 23rd Annual International Symposium on Field-Programmable Custom Computing Machines,pp.68-75,2015.

[3] K. Ito, K.Iizuka, Y.Nu K.Koibuchiand and H. Amano,” Implementing a Multi-ejection Switch and MakinGTHe Use of Multiple Lanes in a Circuit-switched Multi-FPGA System,CANDAR2020,2020.

[4] 稲毛琢己, 弘中和衛, 飯塚 健介,天野英晴：M-KUBOS を用いた PYNQ クラスタの構築,情報処理学会研究会報告 2021-ARC-243,Vol.20,pp.1-6,2021.

[5]M. Yamakur, R.Takano , A.B.Ahmed and H. Amano, ”Development of Multi-tenant Resource Management for Multi-FPGA Cloud Systems,RECONF2020,2020.

[6]Xilinx Inc, ″PYNQ - Python productivity for Zynq - Home}″,http://www.pynq.io/ (accessed 2020-12-21)