NICにおけるPCI Express性能の計測

Yohei Kuga@KEIO Univ

高速PCルータ研究会 2015/5

今日の話

1. PCIe NICの基礎体力測定
2. ざっくばらんにFPGA開発ネタ
   - FPGA+SystemVerilogで合成可能なパケット処理を考える
   - その他のFPGA NIC実装

Motivation

ネットワークを高機能化するためにNICを自由に拡張したい

最近のDC+SW界隈 (Kernel bypass/Unikernels) に比べて,
ネットワークHWはまだまだユーザ拡張性が低い

- 現在のOffload機能はL2~L4パケットヘッダ操作が主流
  - Capsulation, CSUM, TCP など
- Programmable NICの検討は始まっている

”NICがProgrammableであること”が最初の一歩

- ヘッダやFIB操作はProgrammable NICやNPUで実現可能
- ペイロード操作/Interrupt/PCIe/遅延制御などを直接触りたいならFPGA NICが有力

今回のチャレンジ

前回: シングルポート1GE NIC

- Lattice ECP3 versa kit (1GEx2 + PCIe1.1)
- PCIE-TX: PIO write + Write combining, RX: DMA write
- Linux Driver and Timestamp機能

今回: マルチポート10GE NIC

- KC705 (10GEx4 + PCIe gen2 x8) or NetFPGA-SUME
- PCIE: TX: DMA read, RX: DMA write
- 性能目標: 少なくとも10G 2ポートはline rate出したい

FPGA NICのどこらへんが難しいのか

いまのところ **マルチポート** らへんが課題

- Ethernetポートは増やせても使えるPCIe帯域は共通
- だめ回路ではマルチポートで性能がだせない可能性がある

しかしマルチポートNICのPCIe利用帯域の見積もりは難しい

- ネットワークではマルチポート送受信利用が前提だがPCIeは共有
- マルチポートEthernet利用時のTLP送信待ち時間が課題
- マルチポート40GE/100GE NICがきびしい理由を計測から考える

今回は，市販NICでPCIeの現実に使える利用可能帯域を計測

計測のねらい

- Ethernetの性能を最大限出すために，現実的なPCIeの利用可能帯域を探りたい
- 一方で，市販NICはPCIe帯域に余裕を持って設計されているため，計測方法に工夫が必要
- 今回はIntel x520-SR2 (10GE x2)を用いて計測

Intel 82599: Host Interface Features[1]

PCIe Host Interface	PCIe gen2 (2.5GT/s, 5GT/s)
Number of Lanes	x1, x2, x4, x8

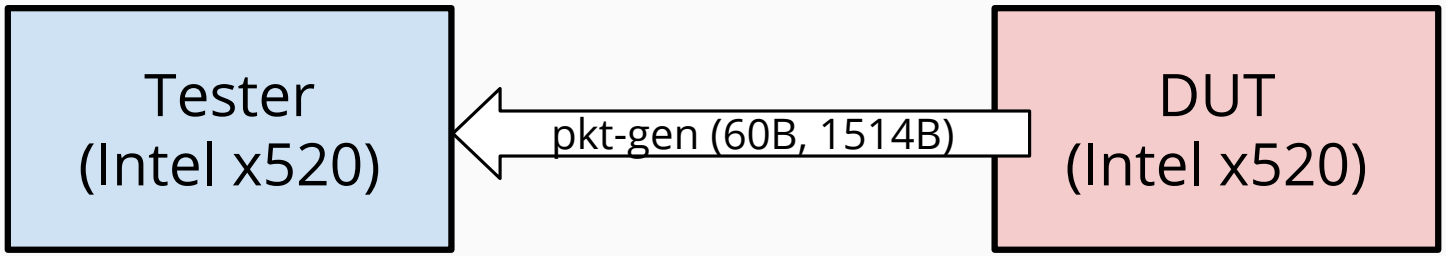
意図的にNIC PCIeの使用レーン数を絞ることでThroughputを計測

[1] (PDF) Intel 82599 10 GbE Controller Datasheet

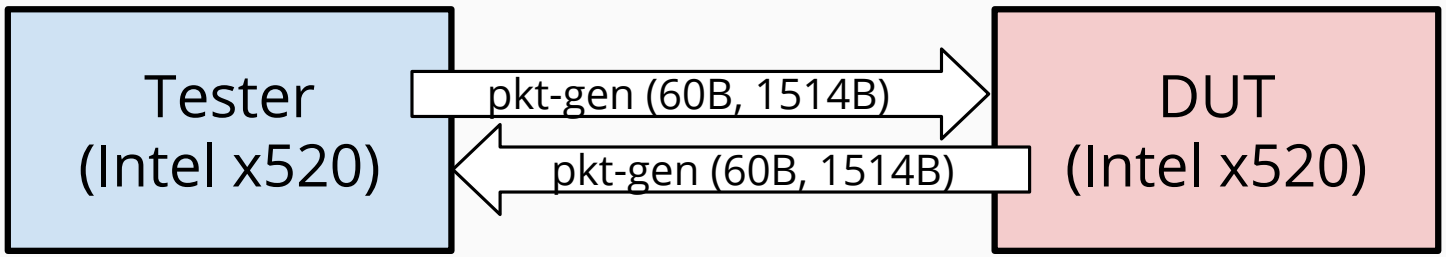
計測環境

Host CPU	Intel Core i7 4770
NIC	Intel x520-SR2
OS	BSD Router 1.55 (FreeBSD 10.1-RELEASE-p8)
Tool	netmap pkt-gen

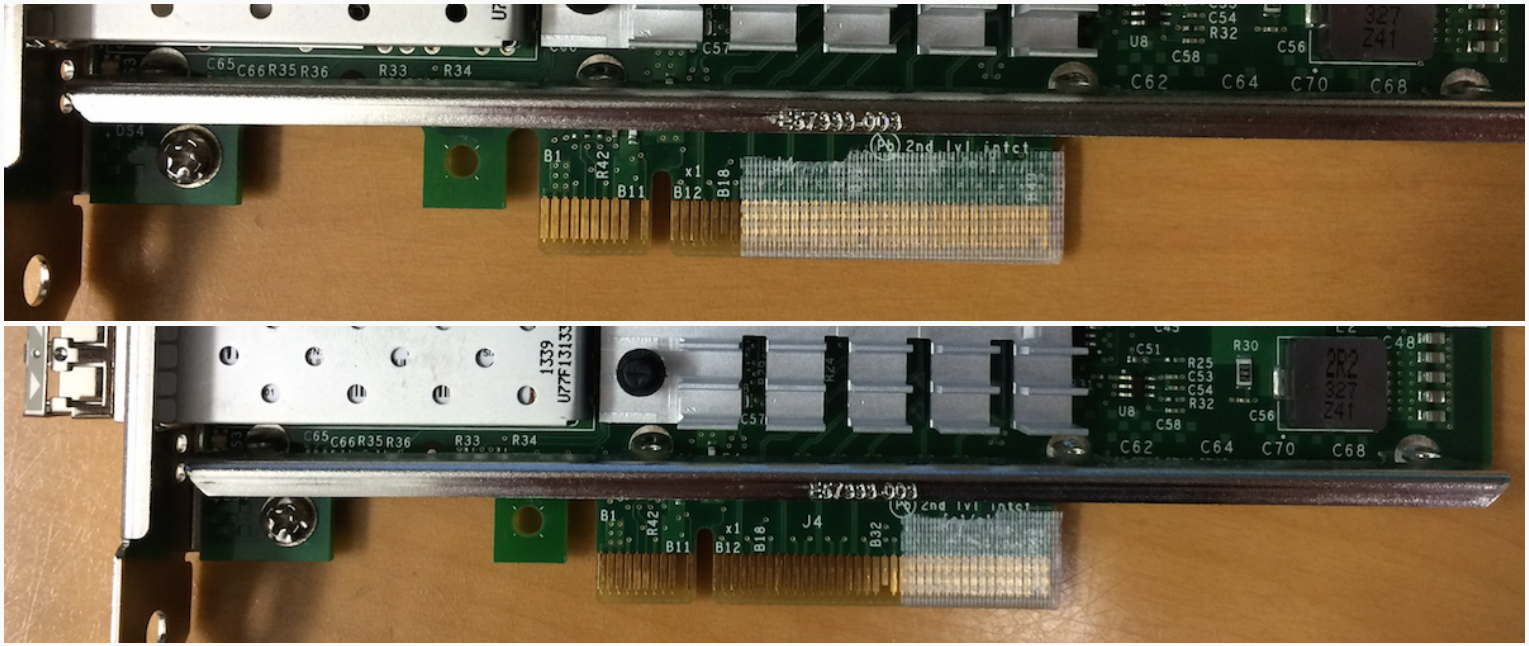
Test 1: TX



Test 2: TXRX (same interface)



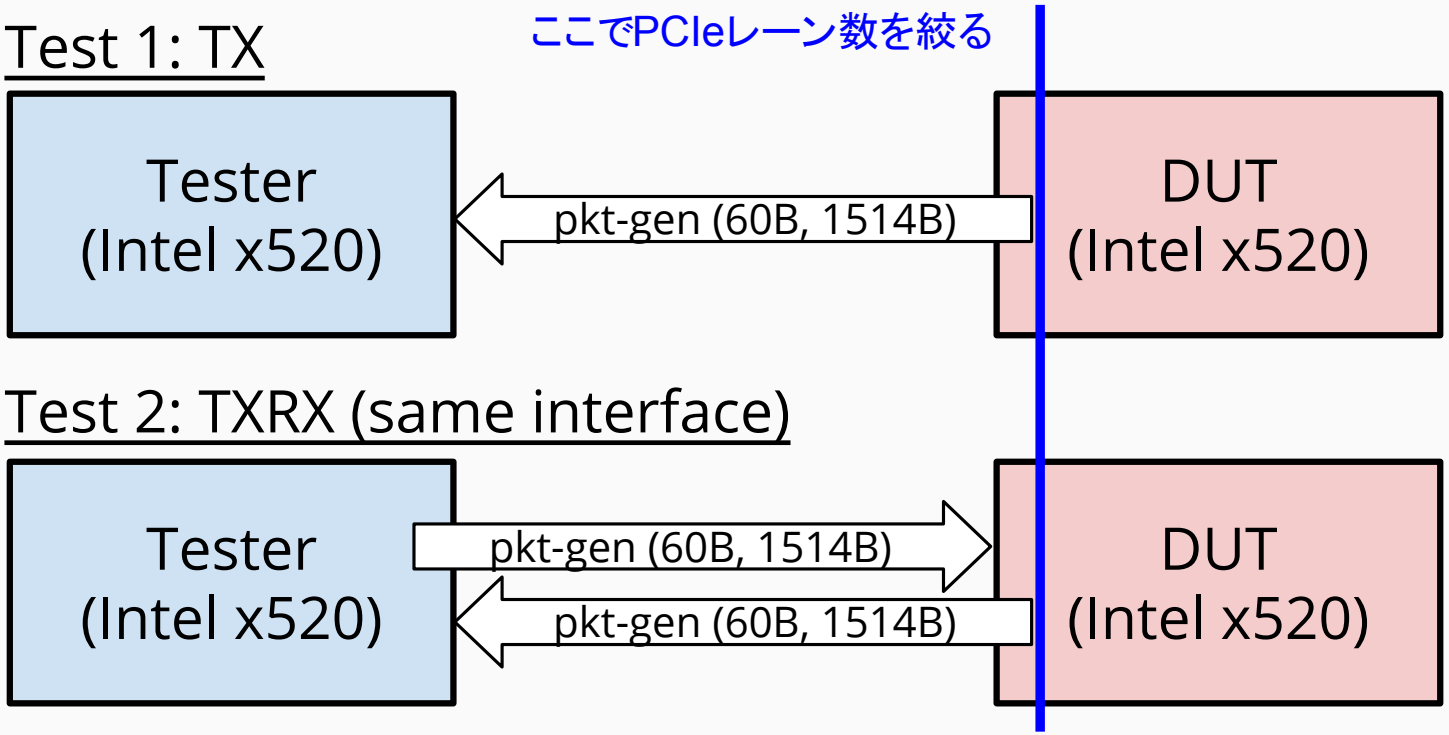
テープを使ってPCIeスロットを物理マスク



```
$ sudo lspci -vv
01:00.0 Ethernet controller: Intel Corporation 82599ES
LnkCap: Port #0, Speed 5GT/s, Width x8, ...
LnkSta: Speed 5GT/s, Width x1, ...
```

上: x1マスク, 中: x4マスク, 下: x1時のLink status

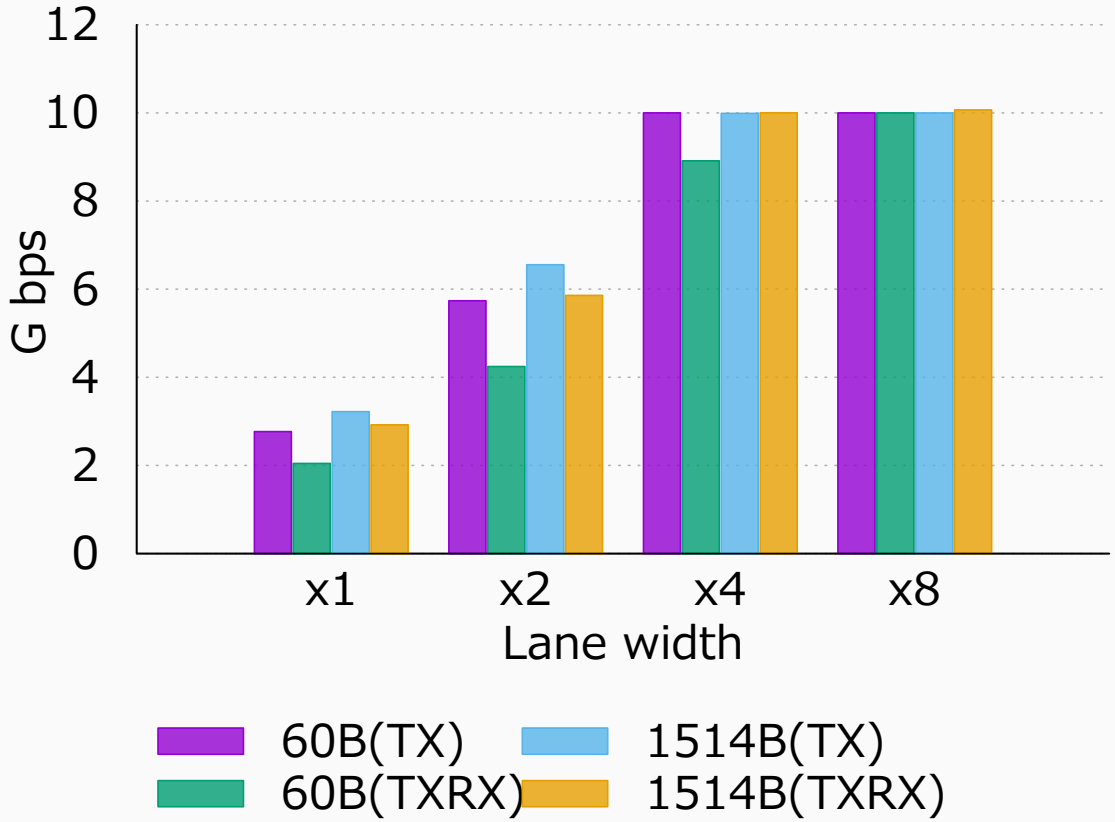
計測環境



計測結果の注意点

- Softwareで試験パケットを生成しているので計測PPSに誤差が生じる
- もちろんx8の時に最大限PCIe性能がだせる回路と考えられるので，NICそのものの絶対評価ではない

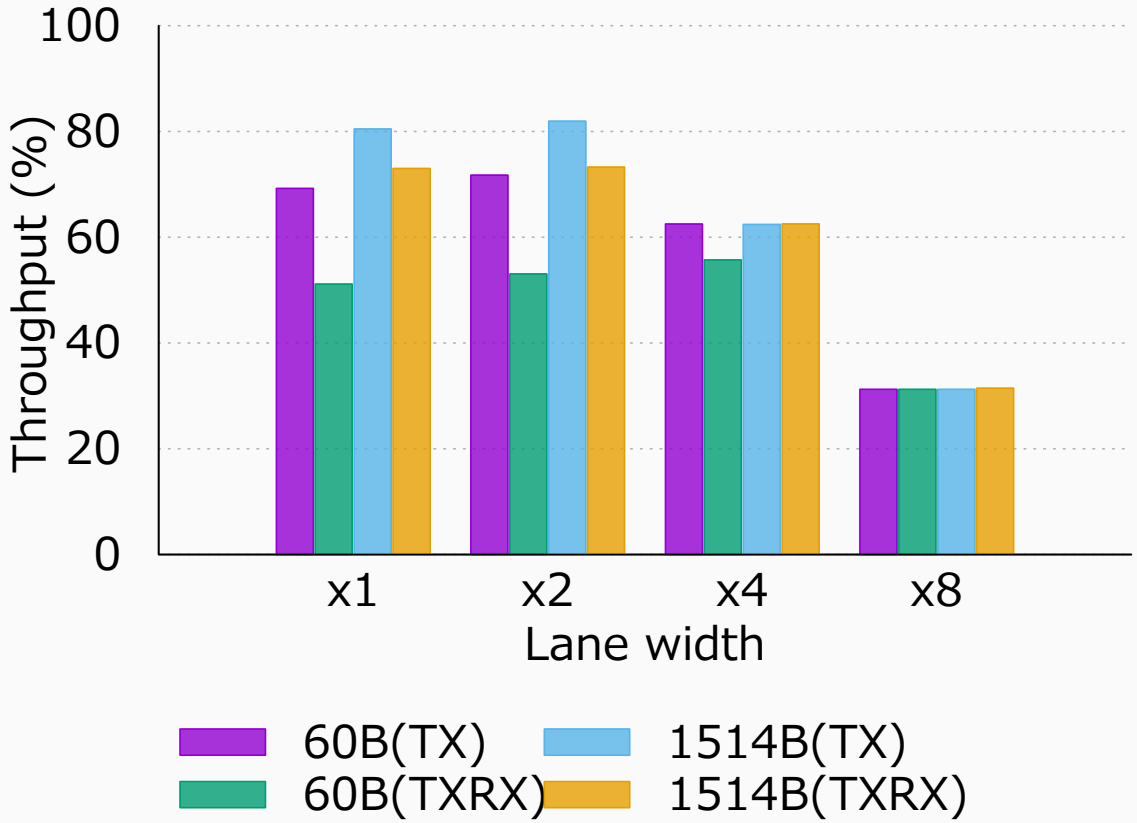
Max. TX Throughput



Theoretical Max. Throughput (GT/s) * 8b/10b overhead (0.8):

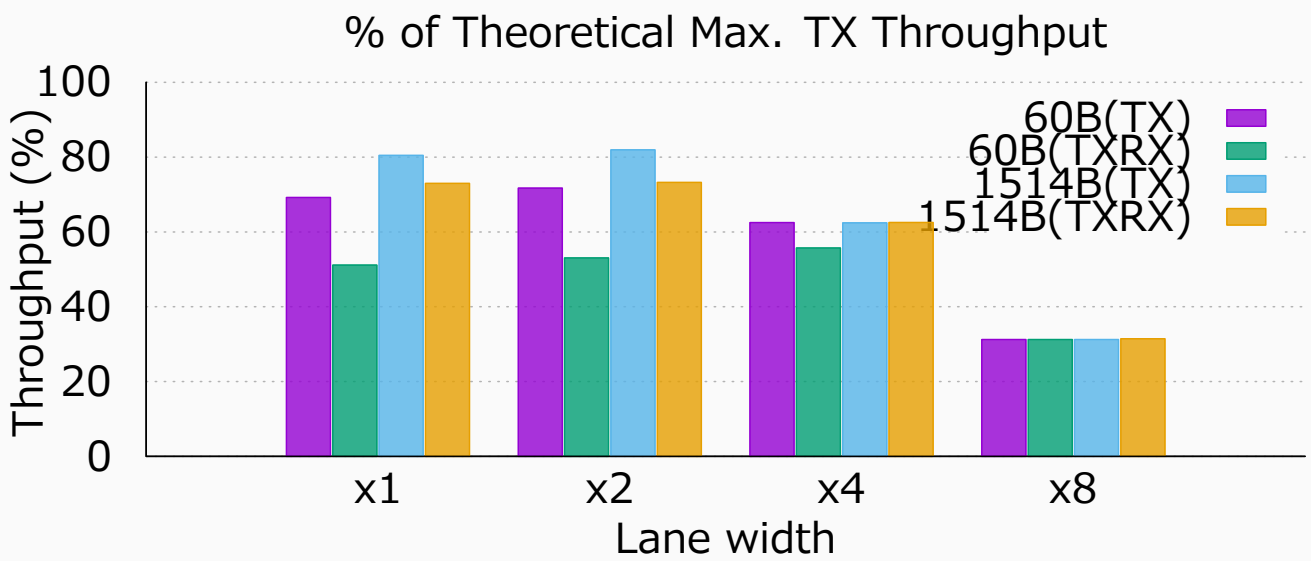
x1: 4Gbps, x2: 8Gbps, x4: 16Gbps, x8: 32Gbps

% of Max. TX Throughput



Theoretical Max. Throughput (GT/s) * 8b/10b overhead (0.8):
x1: 4Gbps, x2: 8Gbps, x4: 16Gbps, x8: 32Gbps

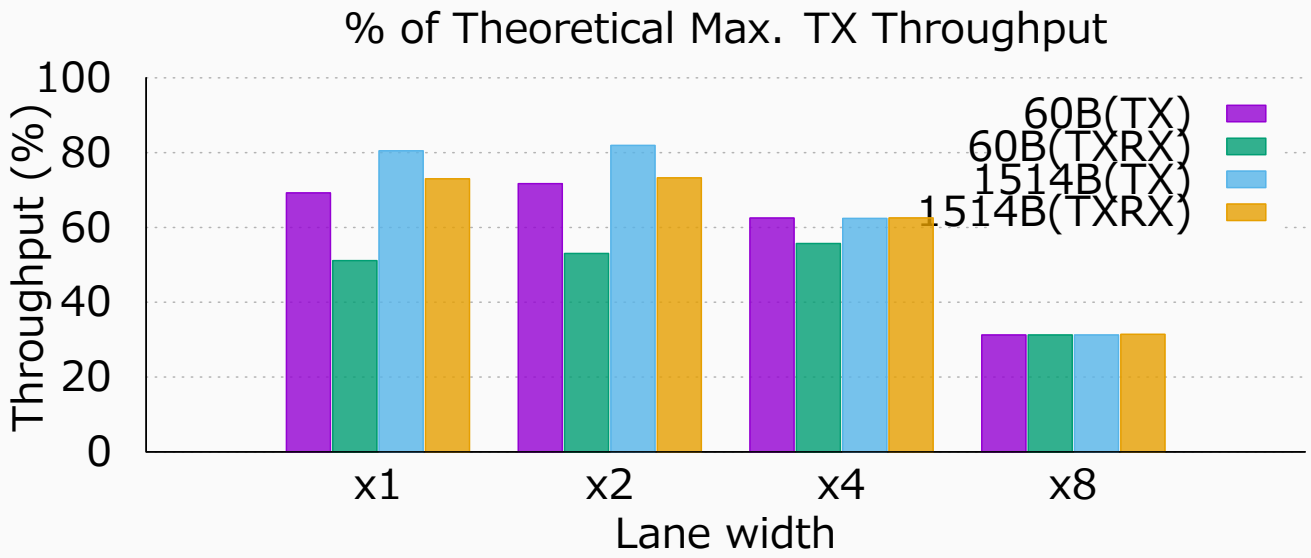
NICのPCIe基礎体力測定 考察



PCIe利用可能帯域を計測することでNIC回路の性能を推測

- 送信のみの場合，PCIeの約81%の帯域が利用可能
- 送受信時，PCIeの約73%の帯域が利用可能
- 送受信時，1514B-60Bで約20%利用可能帯域が減少
- (PCIeはFull duplexにも関わらず) 60B送受信-送信のみで，PCIe利用可能帯域が約18%減少

NICのPCIe基礎体力測定 考察



- PCIeの最大利用可能帯域の割合で見るとx1, x2で同じ傾向
- x4(TXRX60B以外)とx8では，Ethernet帯域(10Gbps)を上回るので特性は見えない

検討内容

1. パケットサイズとPCIe利用可能帯域の関係
2. Full duplexとPCIe利用可能帯域の関係

NICのおさらい

パケット送信時のPCIe操作

1. ドライバがNICのリングバッファのTail pointerを更新
2. データ転送 (DMA read)
3. Write-back the NIC status

パケット受信時のPCIe操作

1. データ転送 (DMA write)
2. ドライバがTail pointerを更新

PCIeのおさらい

- PCIeのメモリ操作は送受信のレーンを使ってTLPパケットの通信が発生
- Ethernetポートを送受信パケットでFull duplex専有しても，PCIeでは上り下りリンクの専有ができるわけではない
  ⇛ TLPパケットの送信待ちが発生する

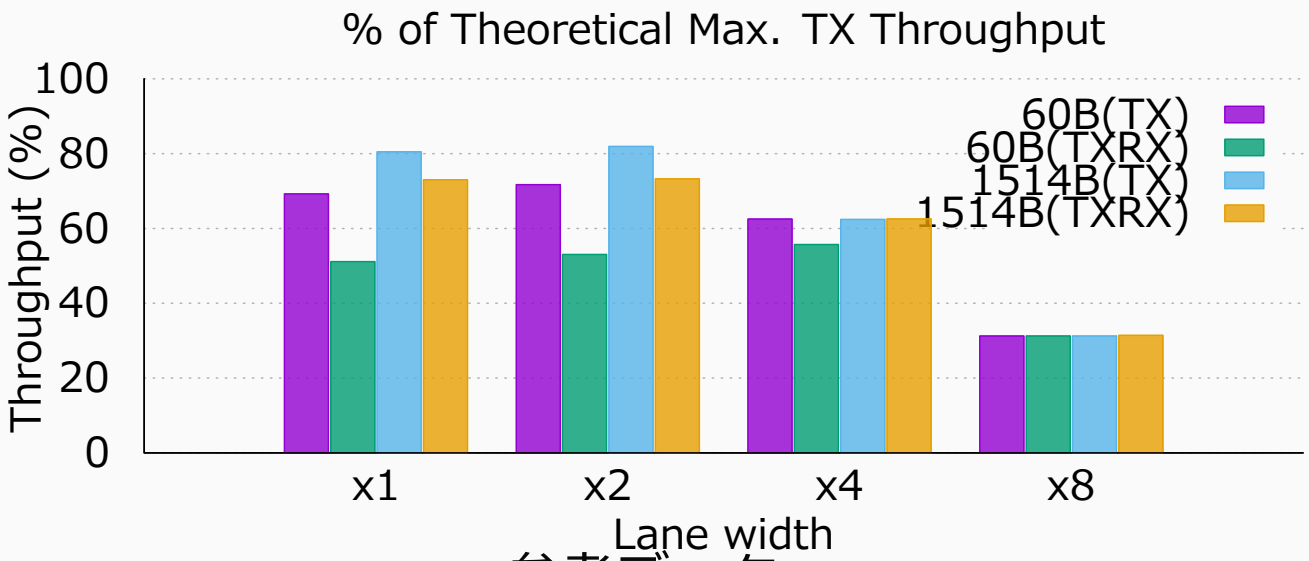
Ethernetパケット送信 (DMA read)

1. TLP (Memory read)を送信
2. ACKとデータ付きコンプリーションを受信
3. コンプリーションに対するACKを送信

Ethernetパケット受信時 (DMA write)

1. TLP (Memory write)を送信
2. ACKを受信

計測結果をどう考えるか



参考データ

	Throughput	%. gen3x8(64Gbps)	%. gen3x16(128Gbps)
10GE	10Gbps	15	8
40GE	40Gbps	62	31
100GE	100Gbps	-	78

Xeon環境があるとPCMでもう少し詳細がわかりそう

NICのレーン数の変更はEEPROMの書き換えで変更できるかも

ざっくばらんにFPGA開発ネタ

FPGA開発環境の話

- Xilinx vivadoではSystemverilogで論理合成が可能になった
  - 使える型が増えた (typedef, union, struct, enum, etc)
    ⇛ FPGA回路の合成時に型のチェックができるだけでもかなりうれしい
  - Classなど合成できないSVの機能はまだまだ多い
  - FPGAでのunionやenumはとても強力
  - VHDL, Verilog混在環境で合成可能
- Systemverilogがとても良かったので合成可能なパケット処理を考えてみる
  - Classなどが使えないのでOSSで公開されているような検証用ライブラリはそのままでは使えない
  - そこでLinuxのRaw socketぽくしてみる

Systemverilogによるパケット処理 [1/3]

ethernet_pkg.sv

```
/* MAC adderss */
typedef bit [ETH_ALEN-1:0][7:0] macaddr_t;

/* ethernet header */
typedef struct packed {
    macaddr_t h_dest;
    macaddr_t h_source;
    bit [15:0] h_proto;
} ethhdr;
```

Systemverilogによるパケット処理 [2/3]

user_app.sv

```
union packed {
    bit [5:0][63:0] raw;        // XGMII (64bit)
    struct packed {
        ethhdr eth;
        iphdr ip;
        udphdr udp;
        bit [47:0] padding;
    } hdr;
} tx_pkt, rx_pkt;
```

Systemverilogによるパケット処理 [3/3]

TX

```
// User register to XGMII_TX
xgmii.data = endian_conv64(tx_pkt.raw[5]);
```

RX

```
// XGMII_RX to User register
rx_pkt.raw[5] <= endian_conv64(xgmii_rx.data);
...
// packet filtering
if (rx_pkt.hdr.eth.h_proto == ETH_P_IP &&
    rx_pkt.hdr.ip.protocol == IP4_PROTO_UDP &&
    rx_pkt.hdr.udp.dest    == 16'd9) begin
```

Raw socket ぽい?

その他のFPGA NIC実装

- NetFPGA-10G (最近1ポート10Gラインレート対応)
- NetFPGA-SUME (リファレンス回路待ち)
- KC705, VC709などのリファレンスNIC

それぞれ論理合成に2から3時間かかる😐

Summary

- 実験用のFPGA NIC回路を作るために，まずは市販NICのPCIe帯域を計測
  - PCIeの使用レーンを減らすことでPCIe利用可能帯域を計測
  - Ethert送受信時はPCIeの利用可能帯域が60%前後まで下がった
  - NICはEthernetの広帯域化に従って，PCIe利用可能帯域の効率化が課題になる
- 100GE NIC以降はNIC間をデイジーチェーンでつないでPCIe帯域を増やすアプローチもあるらしい
- 今後は現在一般的なNICリングバッファ構造以外の方法を検討

Questions? ☺