

# イーサネットコア の使い方

2017/7/25

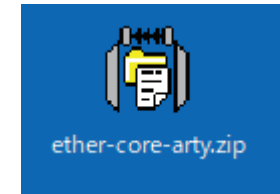
特殊電子回路(株)

For Seccamp 2017

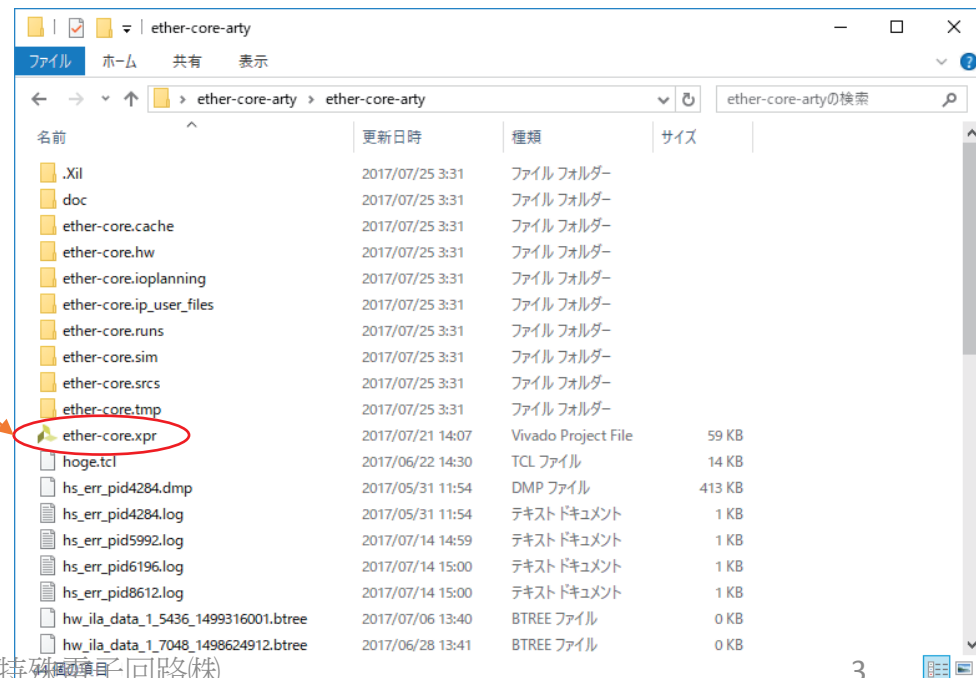
# これは何ですか？

- Arty用に作られたイーサネットコントローラ
  - Arty用は100Mbps版
  - (Artyでは無理だけど) 1Gbpsと10Gbpsも可能
- ARPに応答できます。ARPが送信可能です。
- Pingに応答できます。Ping送信可能です。
- 任意のUDPを送信可能です。
- FPGAが問答無用でパケットを送ります。

# プロジェクトの解凍

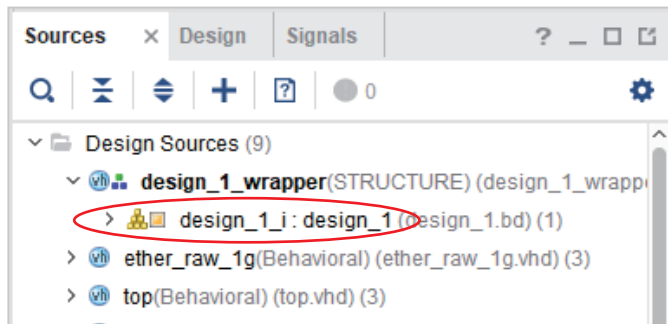


- ダウンロードしたプロジェクトを解凍する
- ether-core-artyフォルダをできるだけ短いディレクトリ名のところ（D:¥直下とか）に移動する
- ether-core.xprをクリック

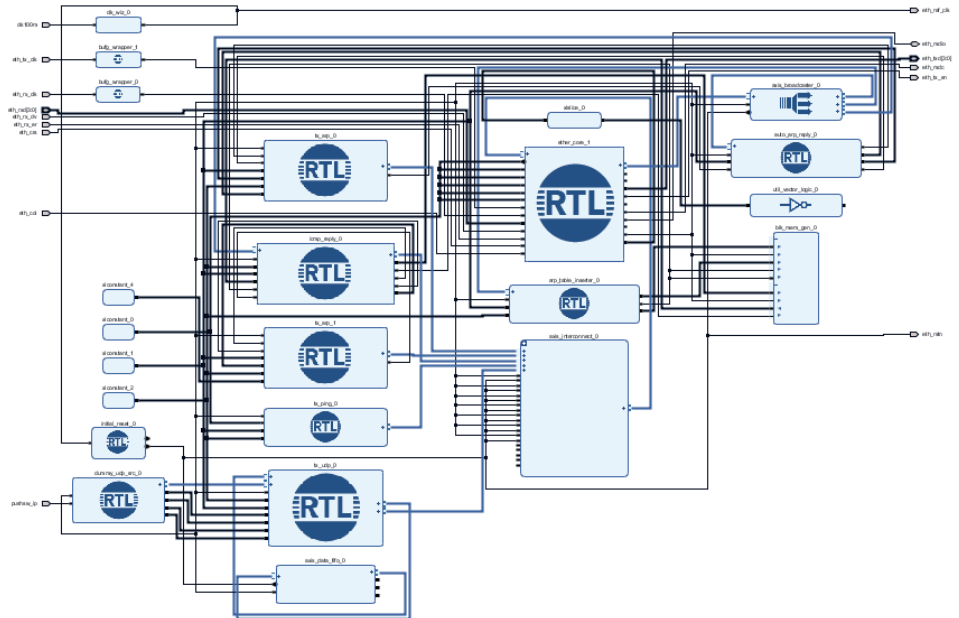


# Vivadoが開く

- design\_1\_wrapperを開いてdesign\_1\_iを開く。

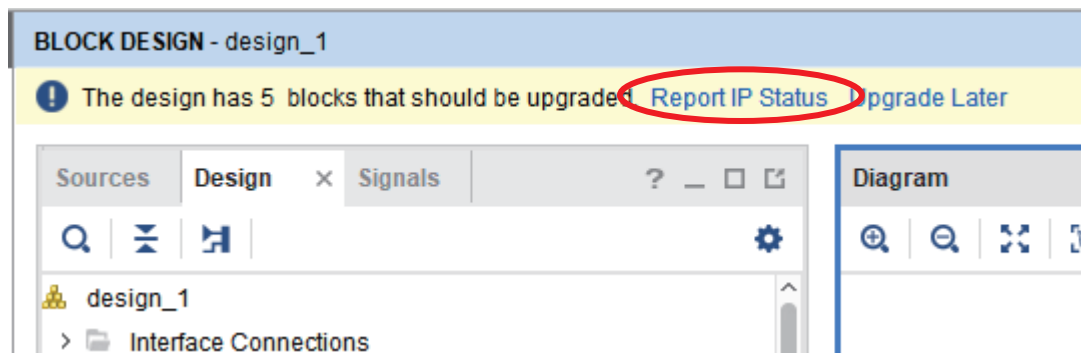


- 構造図(Block Design)が開くので、しばし構造を眺めるべし



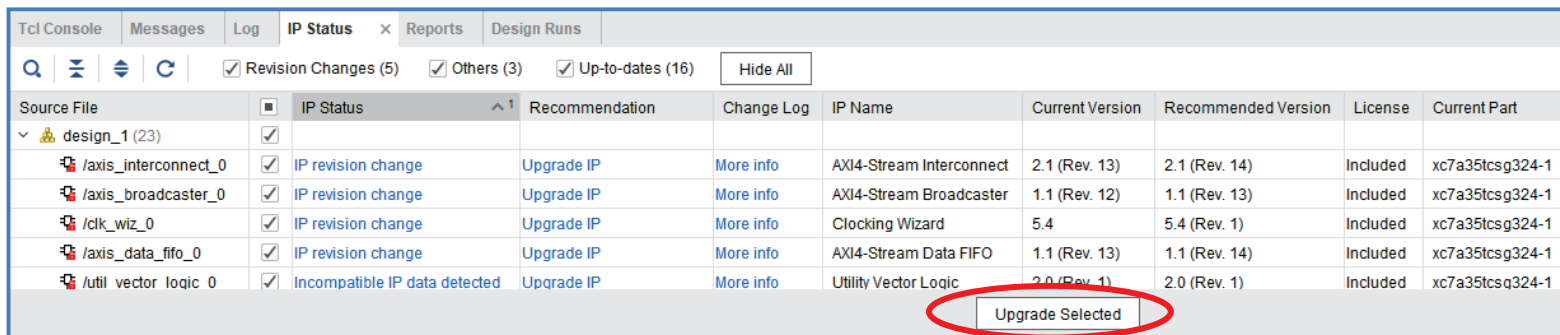
# 開いたら最初にやること

- Vivado 2017.1で作成されたプロジェクトなので、IPコアをバージョンアップしなければならない。
  - ここでいうIPコアとは、サブモジュールみたいなもの。
- まず、Report IP Statusを押す

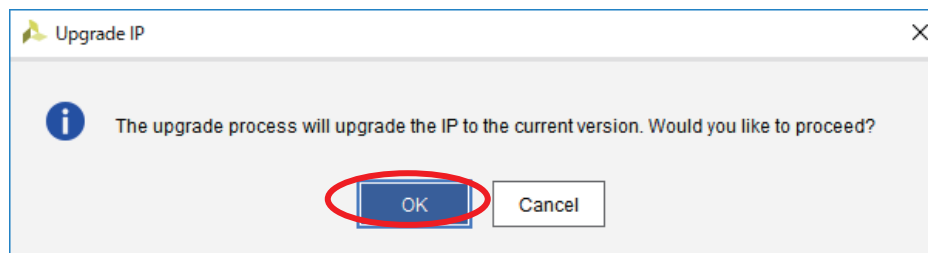


# コアのアップデート

- 下のほうにこういうのが開くので、Upgrade Selectedを押す

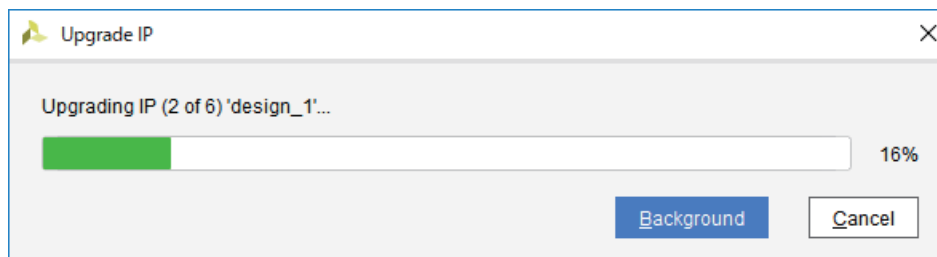
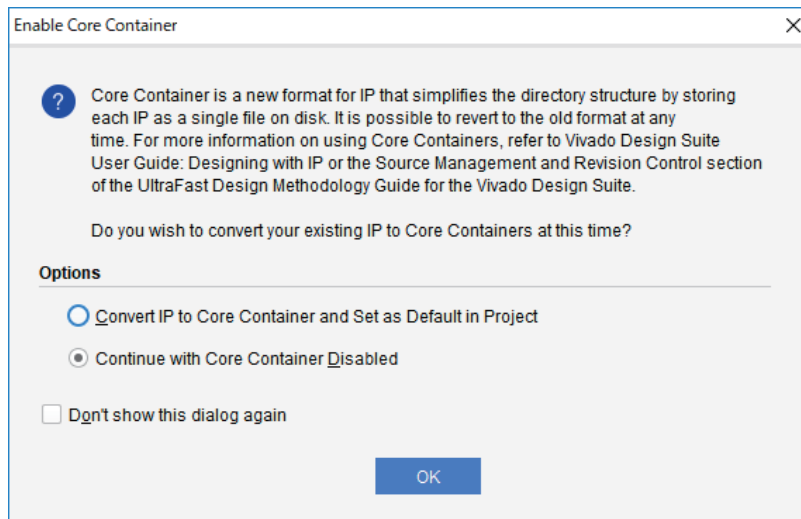


Source File	IP Status	Recommendation	Change Log	IP Name	Current Version	Recommended Version	License	Current Part
design_1 (23)	<input checked="" type="checkbox"/>							
/axis_interconnect_0	<input checked="" type="checkbox"/>	IP revision change	Upgrade IP	AXI4-Stream Interconnect	2.1 (Rev. 13)	2.1 (Rev. 14)	Included	xc7a35tcsq324-1
/axis_broadcaster_0	<input checked="" type="checkbox"/>	IP revision change	Upgrade IP	AXI4-Stream Broadcaster	1.1 (Rev. 12)	1.1 (Rev. 13)	Included	xc7a35tcsq324-1
/clk_wiz_0	<input checked="" type="checkbox"/>	IP revision change	Upgrade IP	Clocking Wizard	5.4	5.4 (Rev. 1)	Included	xc7a35tcsq324-1
/axis_data_fifo_0	<input checked="" type="checkbox"/>	IP revision change	Upgrade IP	AXI4-Stream Data FIFO	1.1 (Rev. 13)	1.1 (Rev. 14)	Included	xc7a35tcsq324-1
/util_vector_logic_0	<input checked="" type="checkbox"/>	Incompatible IP data detected	Upgrade IP	Utility Vector Logic	2.0 (Rev. 1)	2.0 (Rev. 1)	Included	xc7a35tcsq324-1



# コアのアップデート 2

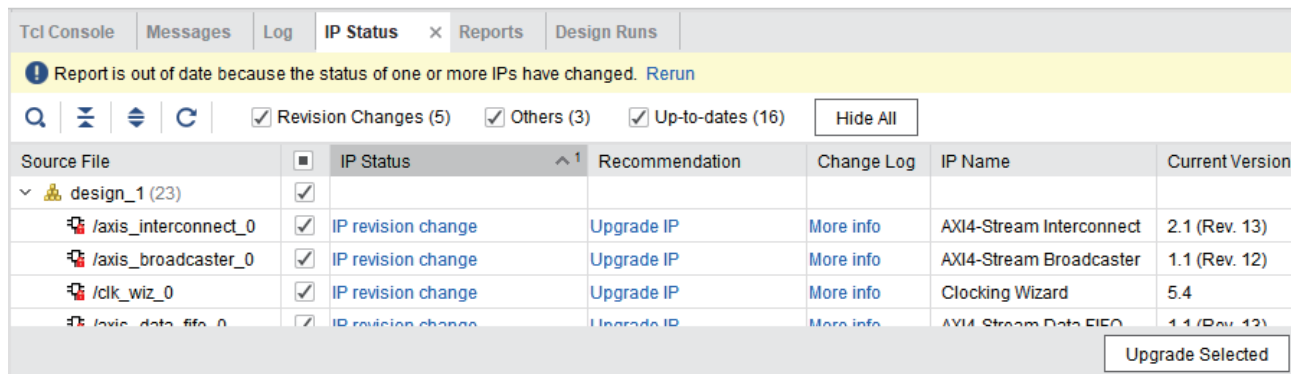
- このダイアログはそのままOKでよい



こういう画面  
ではじっと待つ

# コアのアップデート 3

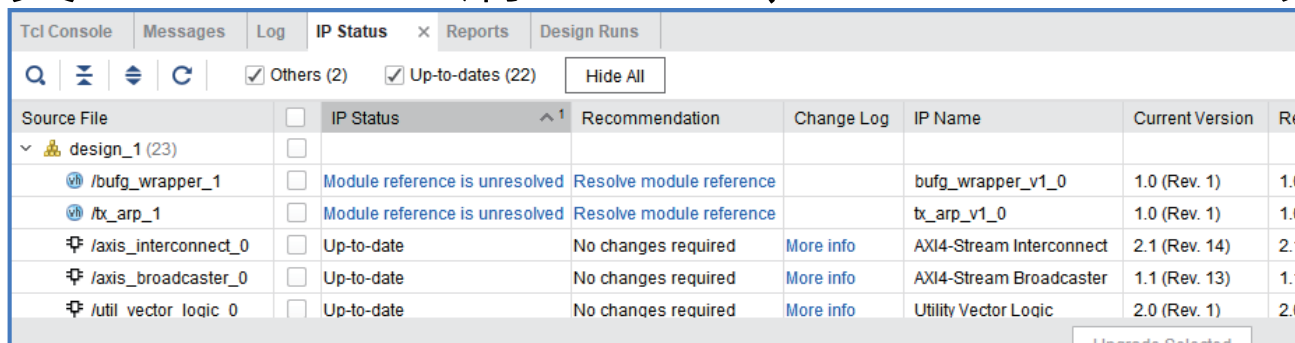
- 処理が終わったら、Rerunを押す



The screenshot shows the IP Status report in a software interface. A yellow banner at the top indicates that the report is out of date because the status of one or more IPs has changed, with a 'Rerun' link. Below the banner, there are search and filter icons, and checkboxes for 'Revision Changes (5)', 'Others (3)', and 'Up-to-dates (16)'. A 'Hide All' button is also present. The main table lists source files and their IP status, recommendation, change log, IP name, and current version. The 'IP Status' column shows 'IP revision change' for several entries, with a recommendation to 'Upgrade IP'. A 'Upgrade Selected' button is visible at the bottom right.

Source File	IP Status	Recommendation	Change Log	IP Name	Current Version
design_1 (23)	<input checked="" type="checkbox"/>				
/axis_interconnect_0	<input checked="" type="checkbox"/> IP revision change	Upgrade IP	More info	AXI4-Stream Interconnect	2.1 (Rev. 13)
/axis_broadcaster_0	<input checked="" type="checkbox"/> IP revision change	Upgrade IP	More info	AXI4-Stream Broadcaster	1.1 (Rev. 12)
/clk_wiz_0	<input checked="" type="checkbox"/> IP revision change	Upgrade IP	More info	Clocking Wizard	5.4
/axis_data_fifo_0	<input checked="" type="checkbox"/> IP revision change	Upgrade IP	More info	AXI4-Stream Data FIFO	1.1 (Rev. 13)

- 黄色いバーが消えたら、アップデート完了

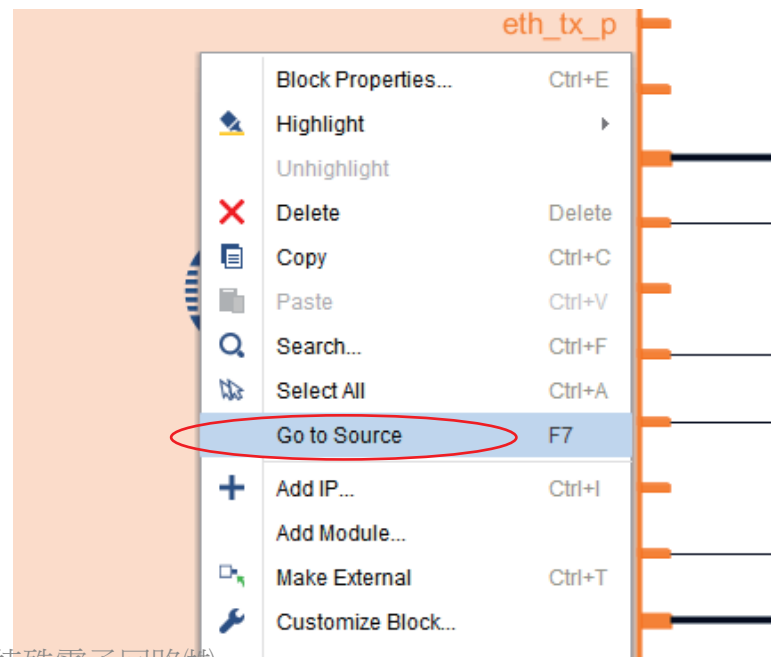
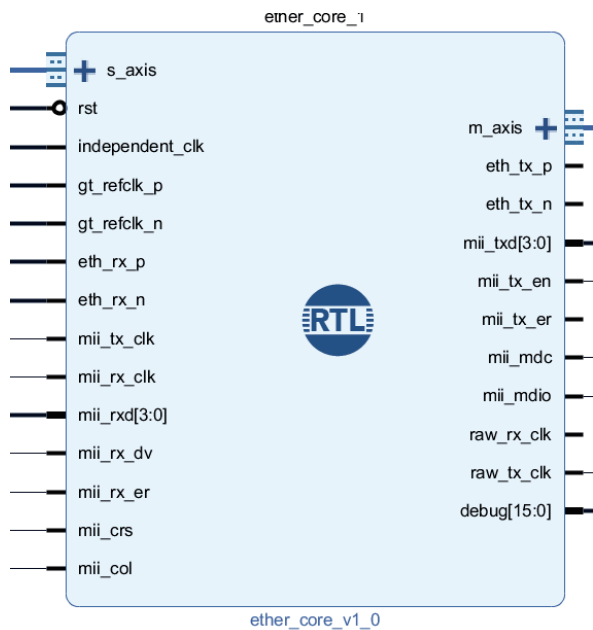


The screenshot shows the IP Status report after the update process. The yellow banner is gone. The 'IP Status' column now shows 'Up-to-date' for several entries, with a recommendation of 'No changes required'. A 'Upgrade Selected' button is still visible at the bottom right.

Source File	IP Status	Recommendation	Change Log	IP Name	Current Version	Re
design_1 (23)	<input type="checkbox"/>					
/bufg_wrapper_1	<input type="checkbox"/> Module reference is unresolved	Resolve module reference		bufg_wrapper_v1_0	1.0 (Rev. 1)	1.0
/tx_arp_1	<input type="checkbox"/> Module reference is unresolved	Resolve module reference		tx_arp_v1_0	1.0 (Rev. 1)	1.0
/axis_interconnect_0	<input type="checkbox"/> Up-to-date	No changes required	More info	AXI4-Stream Interconnect	2.1 (Rev. 14)	2.1
/axis_broadcaster_0	<input type="checkbox"/> Up-to-date	No changes required	More info	AXI4-Stream Broadcaster	1.1 (Rev. 13)	1.1
/util_vector_logic_0	<input type="checkbox"/> Up-to-date	No changes required	More info	Utility Vector Logic	2.0 (Rev. 1)	2.0

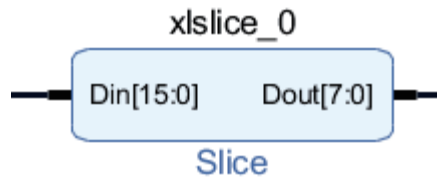
# シンボルの意味 (1)

- RTLと書かれたものは、ハードウェア記述言語で書かれたモジュール。
- 右クリック→go to sourceでソースが見える

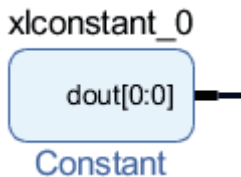


# シンボルの意味 (2)

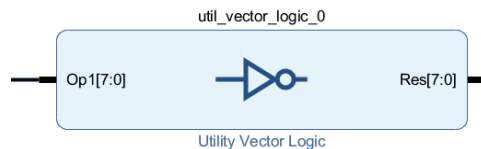
- 小さいのはXILINXが用意した小さなライブラリ



Sliceは、bitを束ねた配線の中から任意の一部を切り出すもの



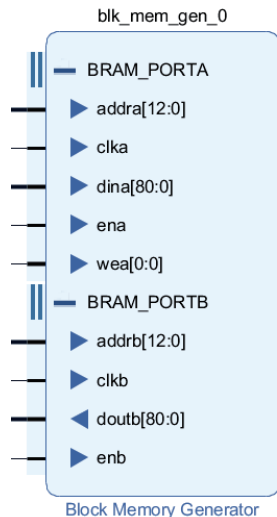
xlconstantは、任意のビット幅で任意の値の定数を作るもの



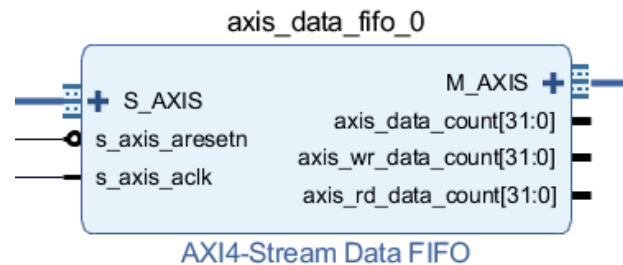
これはNOTゲートが8個束になったもの

# シンボルの意味 (3)

- ちょっと大きめのはIPコア



ブロックメモリ。  
いわゆるSRAM。  
ARPテーブルの格納に使っている。



## AXI StreamデータFIFO

AXIという標準的なバスがあって、  
Streamとはデータを垂れ流すサブ規格。  
このライブラリはそれ用のFIFO。

# Tips (BlockDesignの整理のしかた)

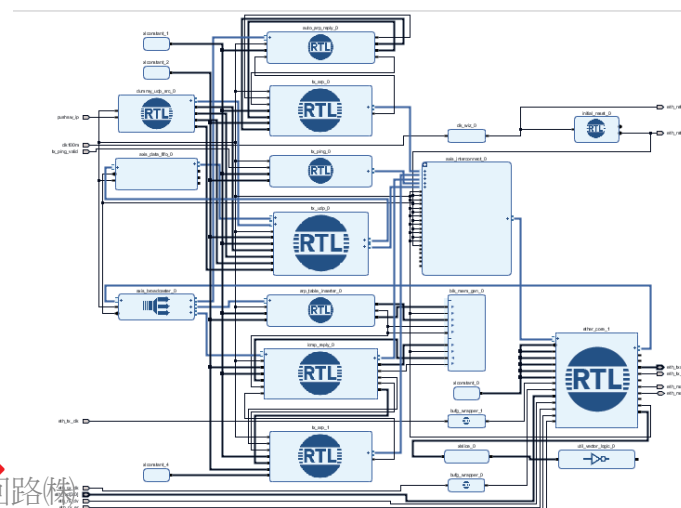
- モジュールを整理したいときは配線の最適化を押す
- モジュールは自分で動かす



配線を最適化する

配置を最適化する (ダメ)

- 配置の最適化を押すとぐちゃぐちゃになってわけがわからなくなるので、押さないこと



配置を最適化した結果→

# AXI Streamって何？

- AXIという規格のStream版です。
- AXIやAXI Streamについては検索するなどして自分で調べてください。
- 本回路では64bit幅です。
- 8バイトを同時に扱うので、それなりにテクニックが必要です。

# モジュールの役割

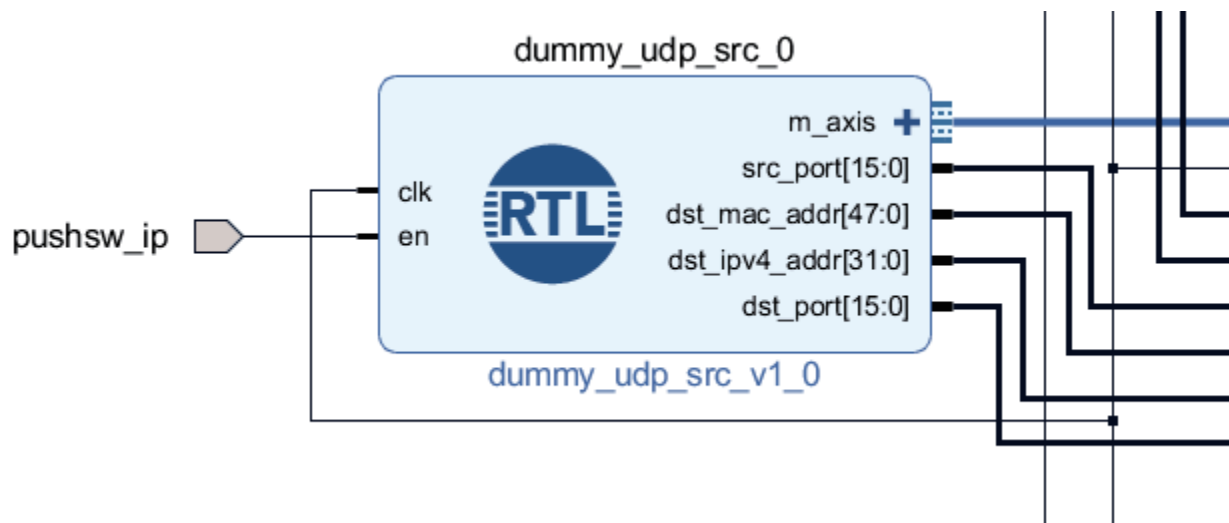
モジュール名	機能	ユーザによる変更
tx_arp_0	ARPリクエストに回答する	×
icmp_reply	PINGに回答する	×
tx_arp_1	PINGに付随するARP応答	×
tx_ping_0	PINGスキャン送信	○
tx_udp_0	UDPのチェックサム計算等	×
dummy_udp_src_0	UDPのパケット送信	○
arp_table_inserter	ARPテーブルの管理	×
ether_core_1	イーサコアの本体	×
auto_arp_reply_0	ARPリクエストに回答する	×
axis_interconnect_0	複数のAXISを束ねる	×

現在のところ、ユーザが変更可能なモジュールは2つです。それ以外のところを変更すると動かなくなる可能性があります。

# ユーザーが編集する回路

# UDPのパケット生成器

- BlockDesignの左下にあるこれ
- 右クリックしてソースを見てください





# UDPのパケット生成

- この回路は、UDPパケットのペイロードとして、00 01 02 03 04 05 06 07 . . . を送信します。
- なぜそうなるかは考えてみてください。
- ここを作りこめば任意のパケットを出せます

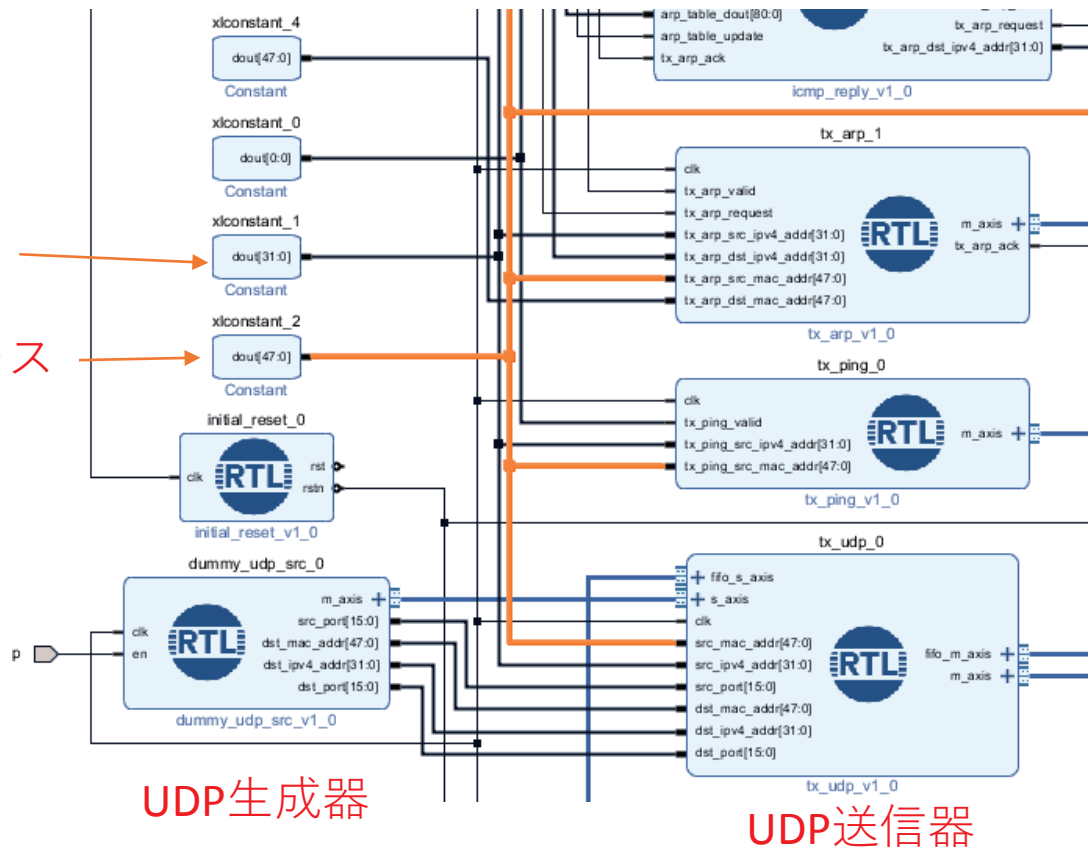
```
when SEND =>
    m_axis_tvalid <= '1';
    m_axis_tdata <= std_logic_vector(data) & std_logic_vector(data + 1) & std_logic_vector(data + 2) &
std_logic_vector(data + 3) & std_logic_vector(data + 4) & std_logic_vector(data + 5) & std_logic_vector(data + 6) &
std_logic_vector(data + 7);
    if m_axis_tready = '1' then
        counter <= std_logic_vector(unsigned(counter) + 1);
        data <= data + 8;
        if unsigned(counter) >= 9 then
            m_axis_tlast <= '1';
            wait_counter <= to_unsigned(INTERVAL, 16);
            data <= (others => '0');
            state <= IDLE;
        end if;
    end if;
```

# ソースアドレスの設定

- xlconstantを突っついて、中身を開く

送信元IPアドレス

送信元MACアドレス

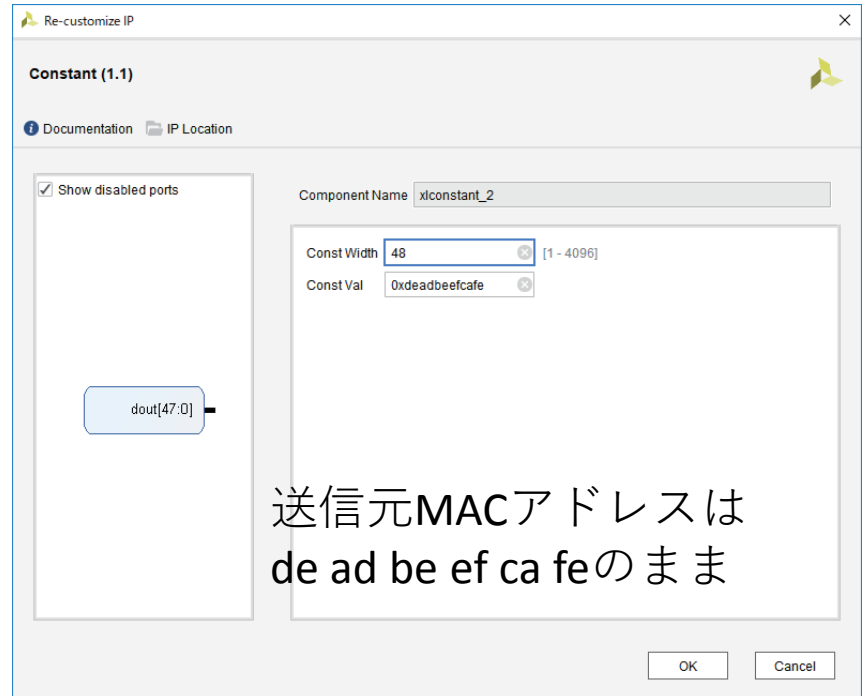
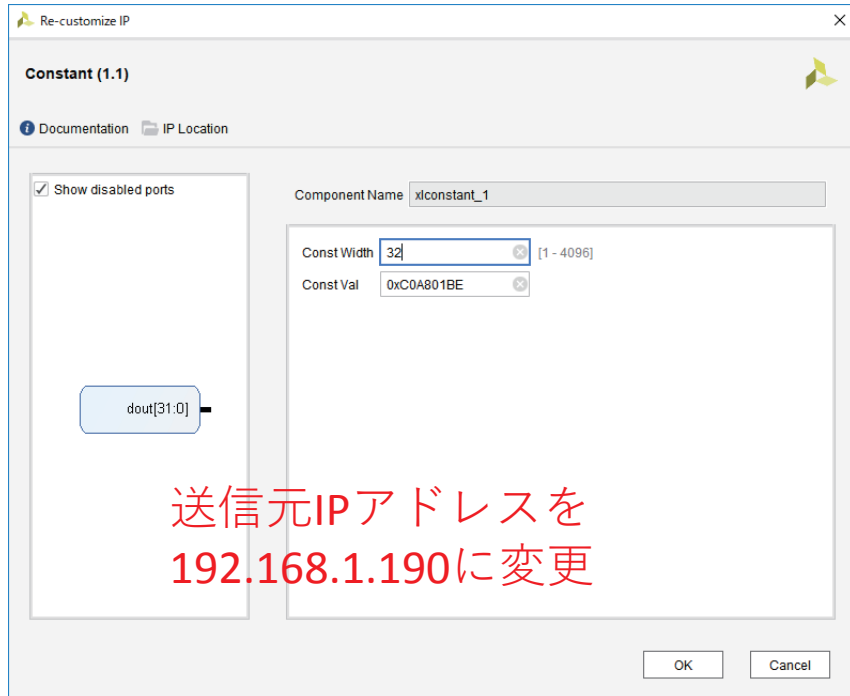


UDP生成器

UDP送信器

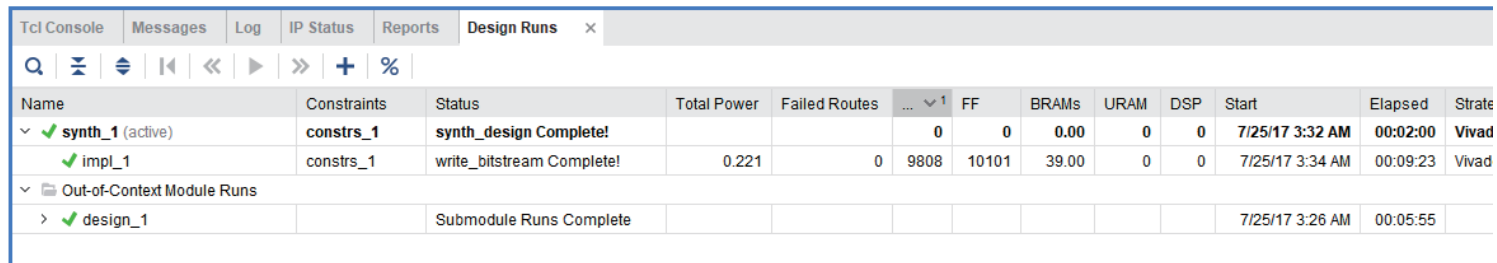
# ソースアドレスの設定 2

- MAC/IPアドレスを16進数で設定する



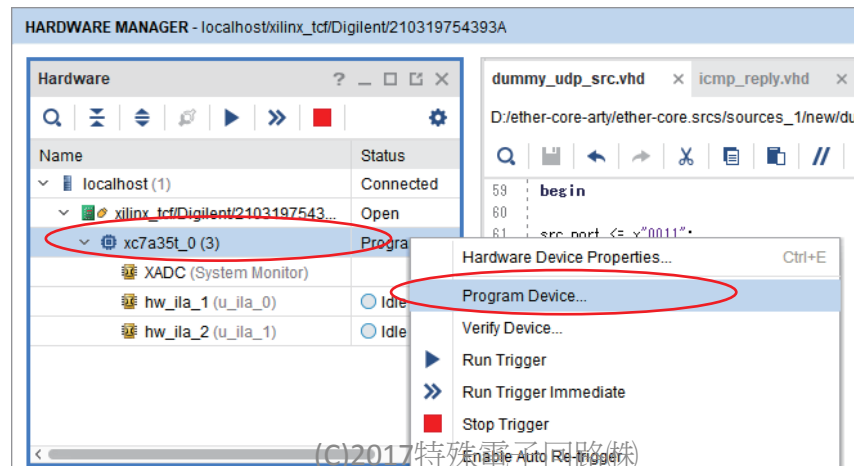
# これで論理合成を開始する

- Core i7のマシンで16分くらいかかります。



Name	Constraints	Status	Total Power	Failed Routes	FF	BRAMs	URAM	DSP	Start	Elapsed	State	
✓ synth_1 (active)	constrs_1	synth_design Complete!			0	0	0.00	0	0	7/25/17 3:32 AM	00:02:00	Vivado
✓ impl_1	constrs_1	write_bitstream Complete!	0.221	0	9808	10101	39.00	0	0	7/25/17 3:34 AM	00:09:23	Vivado
Out-of-Context Module Runs												
> ✓ design_1		Submodule Runs Complete								7/25/17 3:26 AM	00:05:55	

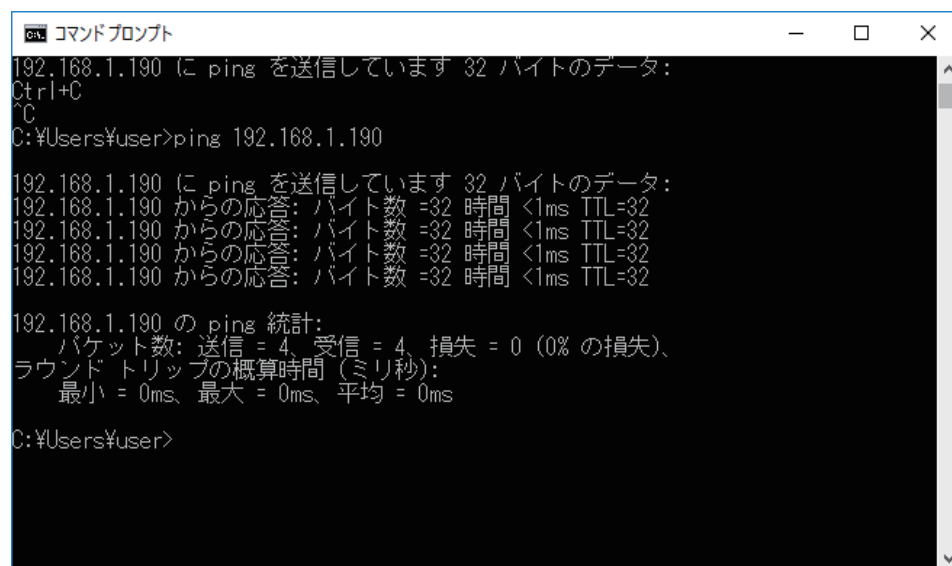
- 合成が終わったら書き込み



# 実験

# ArtyがPINGに回答します

- IPアドレスはFPGA内に作りこんだアドレス (192.168.1.190)です
- 約330usで応答しています



```
コマンドプロンプト
192.168.1.190 に ping を送信しています 32 バイトのデータ:
Ctrl+C
^C
C:\Users\%user>ping 192.168.1.190

192.168.1.190 に ping を送信しています 32 バイトのデータ:
192.168.1.190 からの応答: バイト数 =32 時間 <1ms TTL=32
192.168.1.190 からの応答: バイト数 =32 時間 <1ms TTL=32
192.168.1.190 からの応答: バイト数 =32 時間 <1ms TTL=32
192.168.1.190 からの応答: バイト数 =32 時間 <1ms TTL=32

192.168.1.190 の ping 統計:
    パケット数: 送信 = 4、受信 = 4、損失 = 0 (0% の損失)、
ラウンド トリップの概算時間 (ミリ秒):
    最小 = 0ms、最大 = 0ms、平均 = 0ms

C:\Users\%user>
```

→	2	0.880841	192.168.1.2	192.168.1.190	ICMP	74 Echo (ping) request	id=0x0001, seq
←	3	0.881074	192.168.1.190	192.168.1.2	ICMP	74 Echo (ping) reply	id=0x0001, seq

# 押しボタンBTNOを押すと . .

- 全力でUDPのパケットが送信されます
- 送信間隔は1us程度

The screenshot shows a Wireshark capture of network traffic. The main pane displays a list of packets, with packet 10 selected. The packet list table is as follows:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.1.190	192.168.1.5	UDP	130	17 → 16 Len=88
2	0.000029	192.168.1.190	192.168.1.5	UDP	130	17 → 16 Len=88
3	0.121480	192.168.1.190	192.168.1.5	UDP	130	17 → 16 Len=88
4	4.873178	192.168.1.190	192.168.1.5	UDP	130	17 → 16 Len=88
5	4.873179	192.168.1.190	192.168.1.5	UDP	130	17 → 16 Len=88
6	4.873180	192.168.1.190	192.168.1.5	UDP	130	17 → 16 Len=88
7	4.873180	192.168.1.190	192.168.1.5	UDP	130	17 → 16 Len=88
8	4.873181	192.168.1.190	192.168.1.5	UDP	130	17 → 16 Len=88
9	4.873182	192.168.1.190	192.168.1.5	UDP	130	17 → 16 Len=88
10	4.873182	192.168.1.190	192.168.1.5	UDP	130	17 → 16 Len=88
11	4.873183	192.168.1.190	192.168.1.5	UDP	130	17 → 16 Len=88
12	4.873808	192.168.1.190	192.168.1.5	UDP	130	17 → 16 Len=88
13	4.873809	192.168.1.190	192.168.1.5	UDP	130	17 → 16 Len=88
14	4.873810	192.168.1.190	192.168.1.5	UDP	130	17 → 16 Len=88

Packet 10 details:

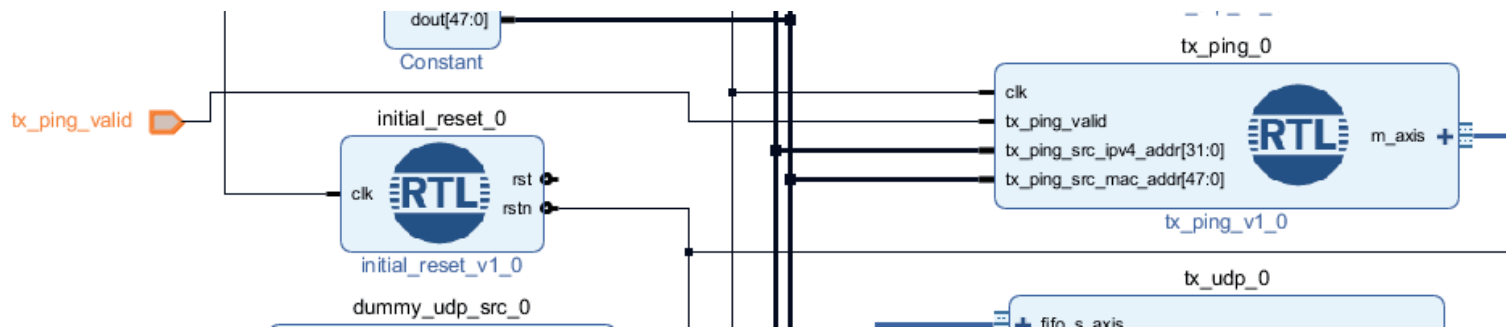
- Frame 10: 130 bytes on wire (1040 bits), 130 bytes captured (1040 bits) on interface 0
- Ethernet II, Src: de:ad:be:ef:ca:fe (de:ad:be:ef:ca:fe), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
- Internet Protocol Version 4, Src: 192.168.1.190, Dst: 192.168.1.5
- User Datagram Protocol, Src Port: 17, Dst Port: 16
- Data (88 bytes)

Hex dump of packet data:

```
0000 ff ff ff ff ff ff de ad be ef ca fe 08 00 45 00 .....E.
0010 00 74 12 34 00 00 20 11 04 32 c0 a8 01 be c0 a8 .t.4...2.....
0020 01 05 00 11 00 10 00 60 50 a3 00 01 02 03 04 05 .....P.....
0030 06 07 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d .....
0040 0e 0f 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d .....
0050 1e 1f 20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d ..!"#$%&'()*+,-
0060 2e 2f 30 31 32 33 34 35 36 37 38 39 3a 3b 3c 3d ./:012345 6789;<=
0070 3e 3f 40 41 42 43 44 45 46 47 48 49 4a 4b 4c 4d >?@ABCDE FGHIJKLM
0080 4e 4f NO
```

# ちょっと改良

- 例えば、tx\_pingモジュールのtx\_ping\_validピンを外に出して、押しボタンBTN1に割り当てます。



## 【XDCファイルの記述】

```
set_property -dict {PACKAGE_PIN C9 IOSTANDARD LVCMOS33} [get_ports tx_ping_valid]
```

# PING爆弾 (スキャン?)

- 192.168.0.0~192.168.255.255まで全アドレスにPINGが送られます。

The screenshot shows a network traffic capture window titled "イーサネットからキャプチャ中". The main pane displays a list of captured packets. The columns are: No., Time, Source, Destination, Protocol, Length, and Info. The packets are all ICMP Echo (ping) replies, with the destination consistently being 192.168.0.11. The source addresses range from 192.40.1.62 to 192.168.1.190. The packet details pane shows the following information:

- Ethernet II, Src: de:ad:be:ef:ca:fe (de:ad:be:ef:ca:fe), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
- Internet Protocol Version 4, Src: 192.168.1.190, Dst: 192.168.0.11
- Internet Control Message Protocol

The hex dump shows the following data:

```
0000 ff ff ff ff ff de ad be ef ca fe 08 00 45 00 .....E.
0010 00 20 00 00 00 00 20 01 8a 75 c0 a8 01 be c0 a8 .....U.....
0020 00 0b 00 00 a8 10 7c fb 00 11 30 31 32 33 34 35 ..012345
0030 36 37 38 39 40 41 42 43 44 45 46 47 6789@ABC DEFG
```

At the bottom, the status bar indicates "Internet Control Message Protocol (icmp), 12 バイト" and "パケット数: 1528 · 表示: 1528 (100.0%)".

みんなも試してみてくださいね！

# 現時点(7/25)のバグ

- たまに、EthernetのFCSの計算が間違っているかも？
- たまに、Pingの応答が悪いことがあるかもしれません。

```
> Frame 3: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
> Ethernet II, Src: de:ad:be:ef:ca:fe (de:ad:be:ef:ca:fe), Dst: [REDACTED]
> Internet Protocol Version 4, Src: 192.168.1.190, Dst: 192.168.1.2
> Internet Control Message Protocol
```

```
0000 [REDACTED] de ad be ef ca fe 08 00 45 00 pM{.A... ..E.
0010 00 30 00 00 00 00 20 01 16 bd c0 a8 01 be c0 a8 .0.... .
0020 01 02 00 00 55 4b 00 01 00 10 61 62 63 64 65 66 ..UK.. ..abcdef
0030 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76 ghijklmn opqrstuv
0040 77 61 62 63 64 65 66 67 68 69 wabcdefg hi
```