Infocube

에뮬레이터 설명서

DU-EMULATOR, 계측기

YM
2020-07-03

목차

Revision History 3

에뮬레이터 개요 4

1. 시작하기 5

1.1 외형소개 5

1.2 시스템 사용준비 (OS : Windows) 6

2. PCAP 파일 재생 7

2.1. 사용법 요약; 7

2.2 사용법 상세 8

2.3 출력파일에 대한 주기 설정 12

2.4 명령어 요약 12

3. Capture Data & PCAP으로 저장 13

3.1 사용법-Capture 13

3.2 저장된 패킷을 '.PCAP'으로 저장 14

4. Terminal 명령어 15

4.1 MAC 변경 15

4.1.1 MAC 변경 활성 15

4.1.1 MAC 변경 비활성 15

4.2 Ping 16

4.3 Fan Control 16

4.4 emul_get_version 17

4.5 emul_get_txinfo_rst 17

4.6 emul_get_txinfo 18

4.7 emul_get_rxinfo_rst 18

4.8 emul_get_rxinfo 18

4.9 emul_get_rxlink_rst_info 19

4.10 emul_get_rxlink_info 19

5. LUA Script 명령 20

5.1. 패킷 카운트 모니터 20

5.2. POE 파워상태모니터 21

6. 사양 & 특성 22

5.1 Player(TX) 22

5.2 TX Packet Timing 22

5.3 Capture(RX) 23

5.4 RX Packet Timing 23

7. Register Map 24

7.1MAC Control Register 24

7.2 Pcap_player Register 24

7.3 Pcap_Capture Register 25

8..... TBD 26

Revision History

Date	Version	Description of Revisions
2020/07/03	1.0	

에뮬레이터 개요

이 에뮬레이터는 다음과 같은 기능을 갖추고 있습니다.

- 10G Ethernet 3+1 Port （1: 광포트）
- PCAP 파일을 4개 포트로 출력
  - Source MAC 설정가능
  - 각 포트 별로 Destination MAC 설정가능
- Packet Capture
  - 4개 각 포트로 수신되는 Uplan Packet을 저장
  - 각 포트 당 1GB 저장 공간
  - 저장된 Uplan Packet을 PCAP 파일로 저장
- Ping
  - eth1
- 외부 입력 10MHz, 1PPS
- 외부 출력 10MHz, 1PPS
- 외부 출력 EVT0, EVT0 (TBD)
- 디버깅 포트 UART
- 디버깅 포트 1000M Ethernet

1. 시작하기

1.1 외형소개

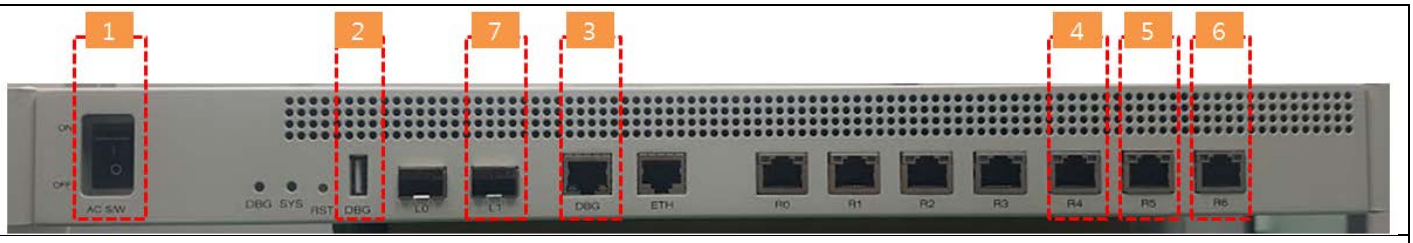


그림1. 전면

1. 전원스위치
2. 디버깅포트 UART
3. 디버깅포트 Ethernet 1G
4. Copper 10G 포트, Port-0
5. Copper 10G 포트, Port-1
6. Copper 10G 포트, Port-2
7. Optical 10G 포트, Port-3

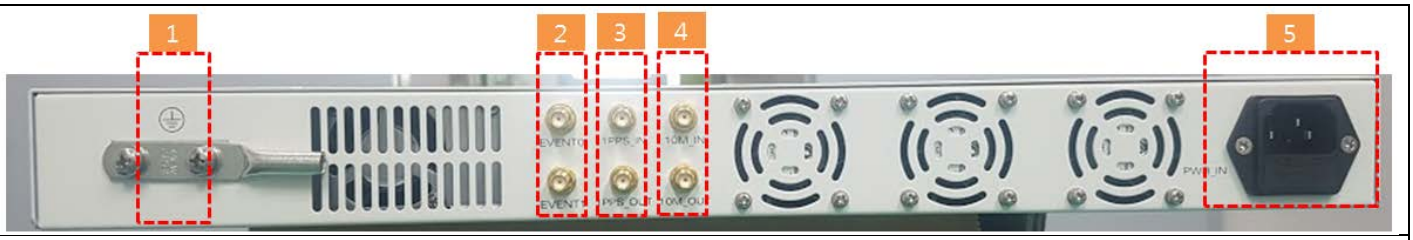
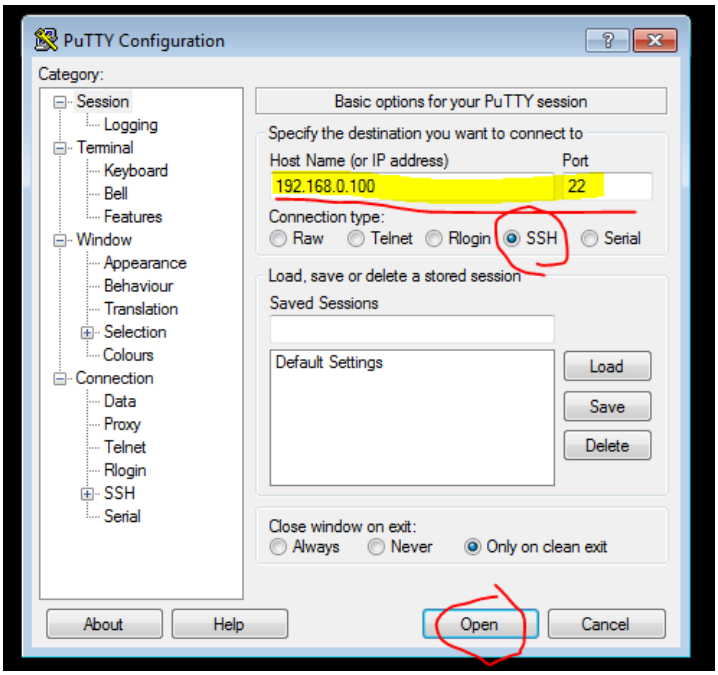


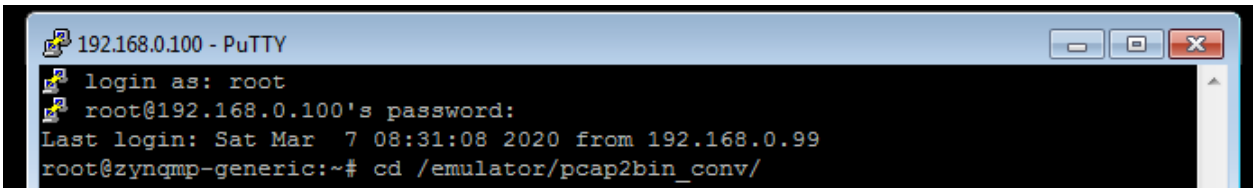
그림2. 후면

1. 새시 그라운드
2. Event Output 2개 포트 (TBD)
3. 1-PPS Input/output Port
4. 10MHz Input/output Port,
   A. 시스템 클럭으로 사용되는 10M-input
   B. 시스템에 동기 된 10M-out
5. 220VAC 전원

1.2 시스템 사용준비 (OS : Windows)

전원을 올리고 부팅이 완료되면, 'PuTTy'와 같은 프로그램으로 아래 그림과 같이 에뮬레이터에 접속한다.
보드 IP는 일반적으로 192.168.0.100~102으로 설정되어 있다.
로그인에 필요한 정보는, id: root passwd : root.





터미널에 'cd /emulator'를 쳐서 이동한다. 에뮬레이터에 '/emulator' 폴더 아래서 모든 작업이 이루어진다.

2. PCAP 파일 재생

2.1. 사용법 요약;

1. '**Putty**', '**Filezilla**', '**Wireshark**'를 PC(**windows**)에 설치한다.
2. **Wireshark**를 사용하여, '**pcap**' 파일을 '**pcapng**' 파일로 변환한다.
3. **Filezilla**를 사용하여, '**pcapng**' 파일을 계측기로 복사한다.
4. **Putty를 사용하여 계측기에 접속**한다.
   A. '**pcapng**' 파일을 **bin**파일로 변경한다.
   B. bin파일을 실행시킨다.

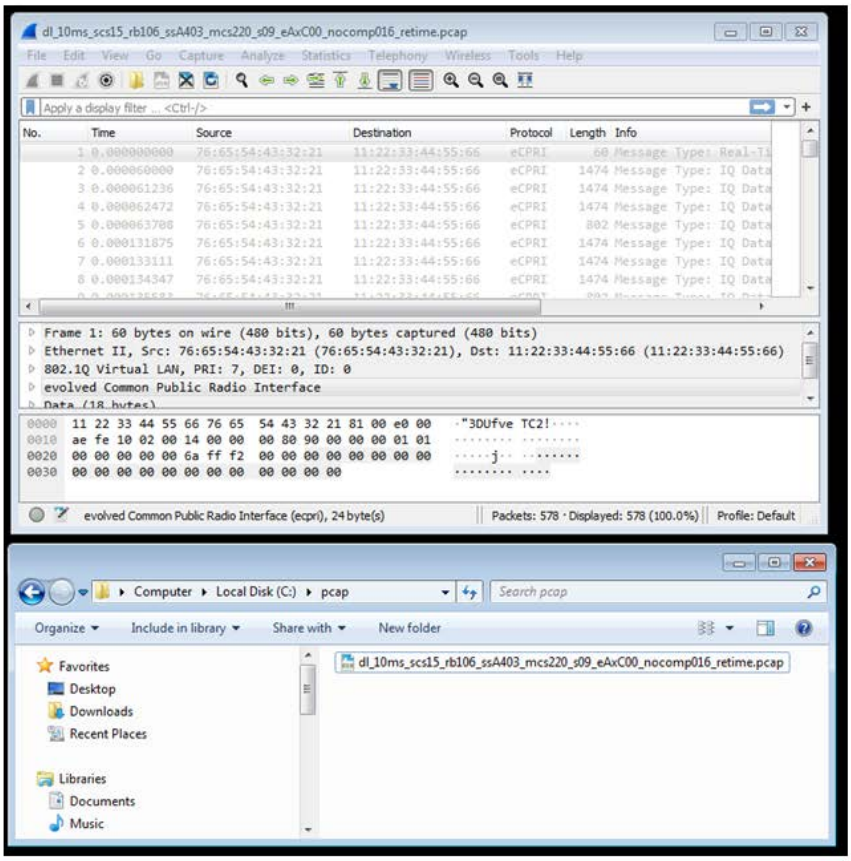
→ 4개 포트에서 데이터가 출력된다.

 # putty로 보드에 접속하여 pcap 파일을 재생하는데 필요한 실행하는 명령어 요약

```
cd /emulator/util
./conv-pcapng-axi128nsec.py    ./full_band_test_20ms.pcapng ./ full_band_test_20ms.bin 0 0
cp -rf    ./full_band_test_20ms.bin /emulator
cd /emulator
./emul_pcap_play_20ms    ./full_band_test_20ms.bin
```

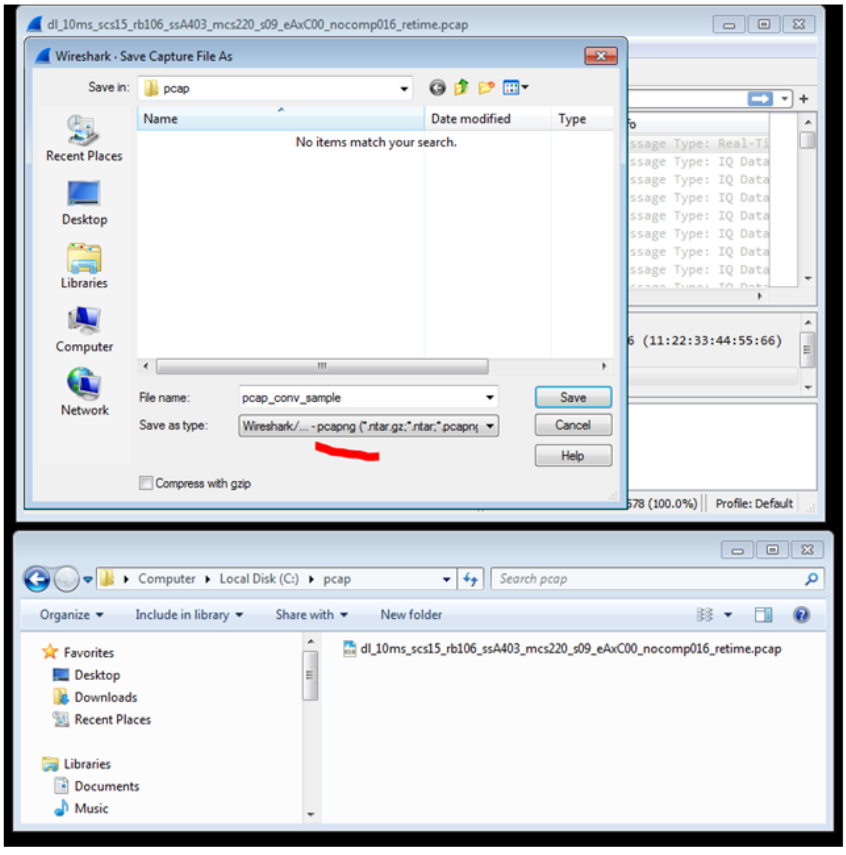
2.2 사용법 상세

PCAP 파일을 PCAPNG로 변경



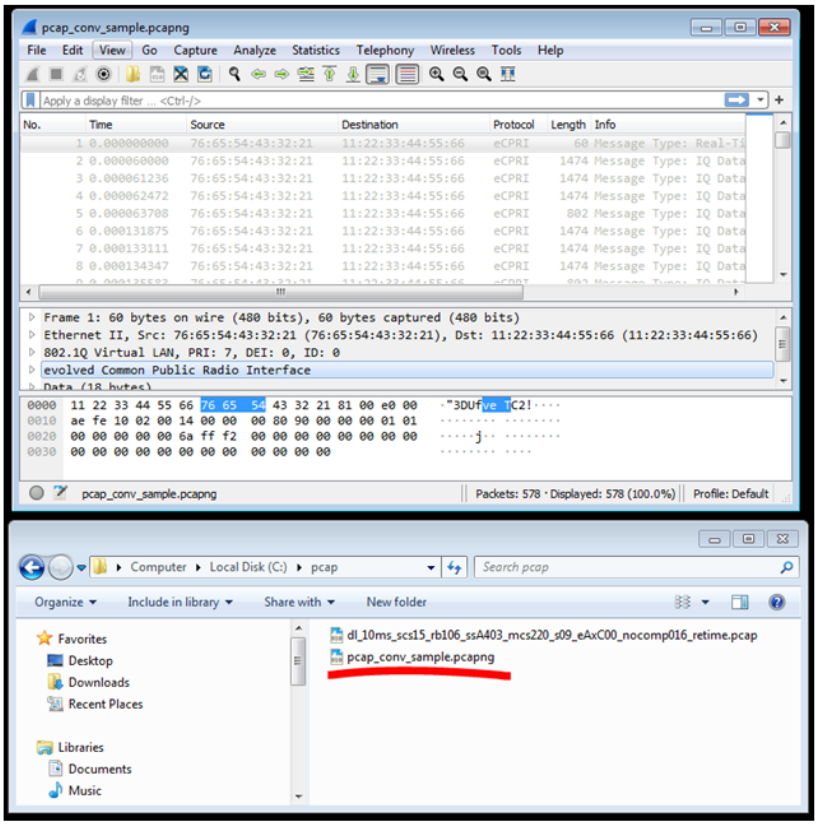
Wireshark로 'PCAP'파일을 연다.

PCAP 파일을 PCAPNG로 변경



Wireshark로 'PCAP'파일을
'pcapng'파일로 다시 저장한다.

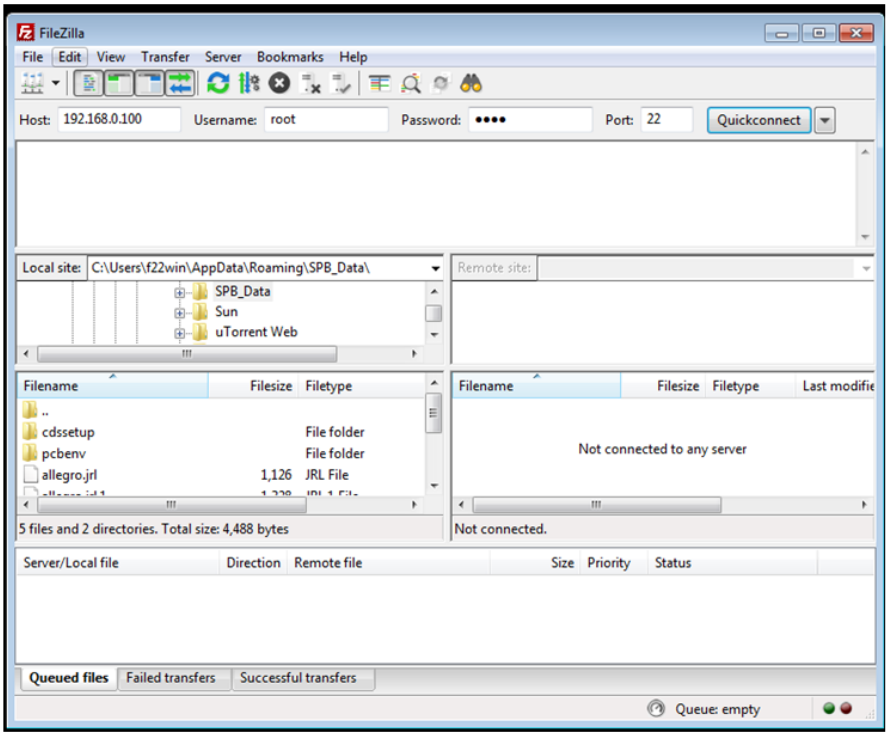
PCAP 파일을 PCAPNG로 변경



그림과 같이 'pcapng' 파일이 생긴다.

pcap_conv_sample.pcapng 파일을 보드에 복사

보드에 원격 로그인(ssh)

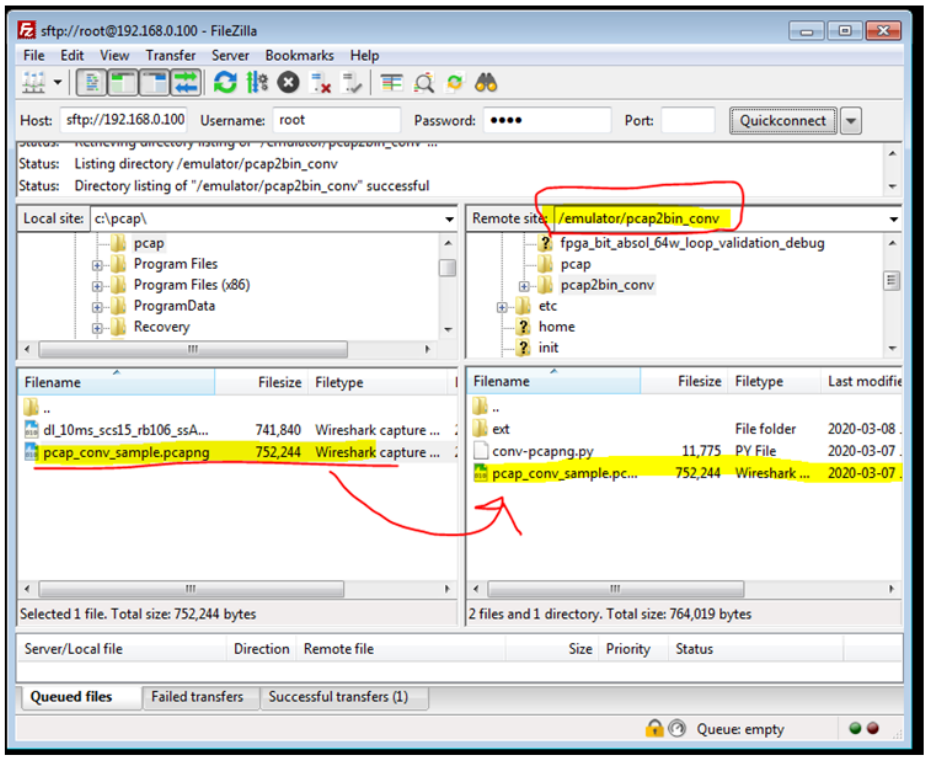


FileZilla를 사용하여,
그림과 같이 아래 정보를 타이핑하고,
'Quickconnect' 버튼을 눌러 접속한다.

Host: 192.168.0.100
Id: root
Pass: root
Port: 22

pcap_conv_sample.pcapng 파일을 보드에 복사

.pcapng 파일을 보드로 복사 (PC폴더에서 -> 보드폴더 / emulator/util)

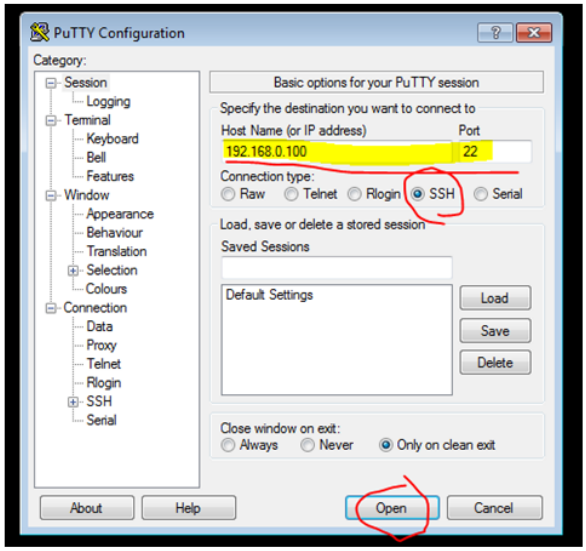


그림과 같이 FileZilla를 사용하여 ,
'pcap'에서 변환한 'pcapng'파일을
보드-폴더에 복사한다.

'pcapng'파일이 복사되는 폴더는,

/emulator/util

Putty로 보드에 접속

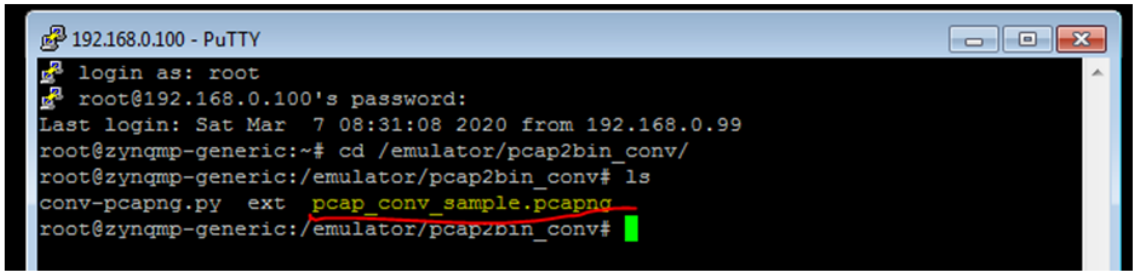


그림과 같이 Putty접속하면, 아래그림과 같이 터미널이 올라온다.

Login : root
Passwd : root
→ 로그인 하고, pcapng 파일이 있는 곳으로 이동한다.

로그인 완료하고,
cd /emulator/ util

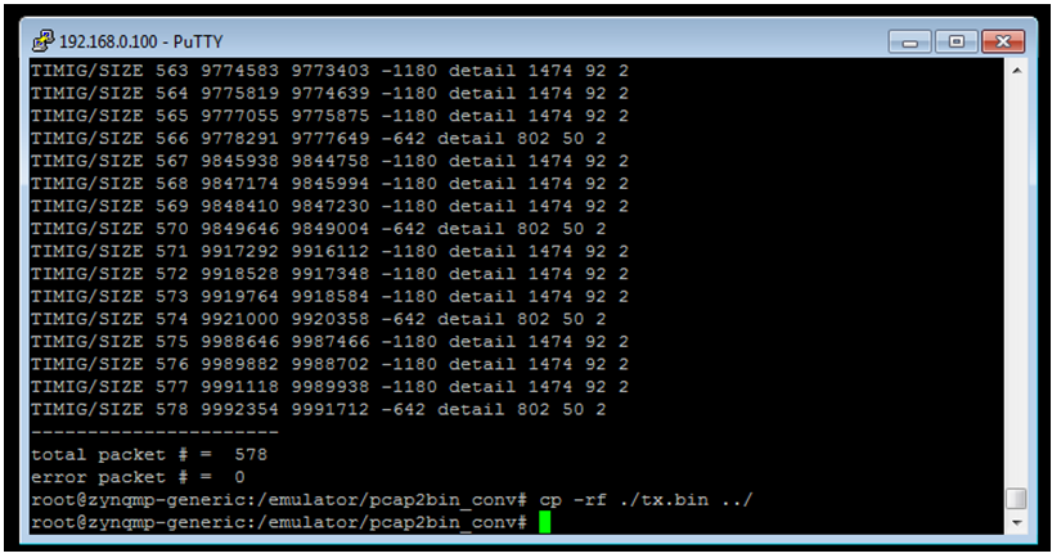
그림과 같이 보드에 복사가 잘됐다면,
'pcap_conv_sample.pcapng' 파일이 있다.



Pcapng 파일을 .bin 파일로 변환

Pcap_player는 pcapng파일을 bin으로 변환해서 사용한다.

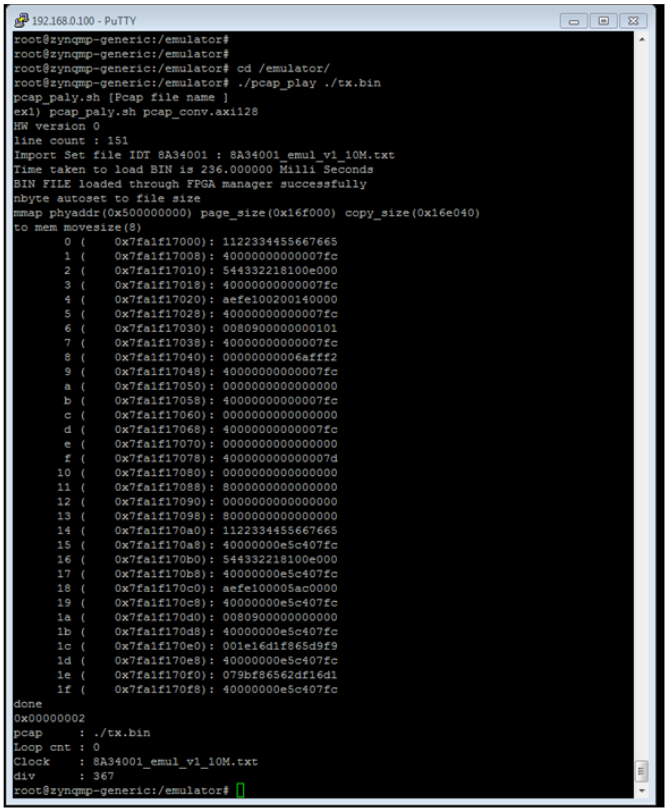
cd /emulator/util
./ conv-pcapng-axi128nsec.py ./pcap_conv_sample.pcapng ./tx.bin 0 0
cp -rf ./tx.bin /emulator



tx.bin 파일 실행

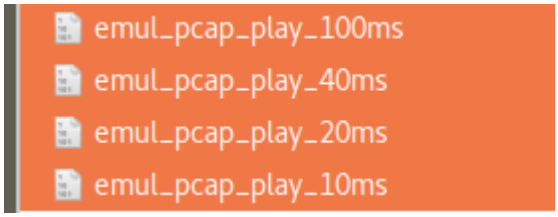
cd /emulator
./ emul_pcap_play ./tx.bin

여기까지 진행하면,
출력데이터가 계속 나간다.



2.3 출력파일에 대한 주기 설정

'/emulator' 안에 아래 그림에 파일이 있고, 해당 시간에 맞는 패턴파일을 재생하면 된다.



➔ 그 외에 시간이 필요한 경우, 지원해드립니다.

2.4 명령어 요약

```
#!/bin/sh


#########################################################################
##  와이어샤크 PCAP을 du-emulator로 출력하는 방법
##  1. 와이어샤크를 사용하요 '.pcap' 파일을  '.pcapng'
#########################################################################


#########################################################################
##  2. 'pcapng'파일을 bin 파일로 변환. bin 파일은 du-emulator용 출력 파일.
#########################################################################
telnet이나 ssh로 (윈도우에서는 푸티나, 파일질라를 이용) 192.168.0.101 또는 192.168.0.102 로 접속한다.
터미널이 열리면 아래와 같이

# 변환라이브러리가 있는 폴더로 이동
# pcapng 파일을 아래와 같이 bin으로 변환
cd /emulator/util
./conv-pcapng-axi128nsec.py ./Pattern_1_full_band_test_20msec.pcapng ./p1_full_band_test_20m.bin 0 0
./conv-pcapng-axi128nsec.py ./Pattern_2_full_band_test_20msec.pcapng ./p2_full_band_test_20m.bin 0 0
./conv-pcapng-axi128nsec.py ./Pattern_3_full_band_test_20msec.pcapng ./p3_full_band_test_20m.bin 0 0
./conv-pcapng-axi128nsec.py ./Pattern_4_full_band_test_20msec.pcapng ./p4_full_band_test_20m.bin 0 0
./conv-pcapng-axi128nsec.py ./Pattern_5_full_band_test_20msec.pcapng ./p5_full_band_test_20m.bin 0 0
./conv-pcapng-axi128nsec.py ./Pattern_6_full_band_test_20msec.pcapng ./p6_full_band_test_20m.bin 0 0
./conv-pcapng-axi128nsec.py ./Pattern_7_full_band_test_20msec.pcapng ./p7_full_band_test_20m.bin 0 0
./conv-pcapng-axi128nsec.py ./Pattern_8_full_band_test_20msec.pcapng ./p8_full_band_test_20m.bin 0 0

# pcap v플레이어가 있는 폴더로 bin파일 복사
cd /emulator/util
cp -f ./p1_full_band_test_20m.bin /emulator
cp -f ./p2_full_band_test_20m.bin /emulator
cp -f ./p3_full_band_test_20m.bin /emulator
cp -f ./p4_full_band_test_20m.bin /emulator
cp -f ./p5_full_band_test_20m.bin /emulator
cp -f ./p6_full_band_test_20m.bin /emulator
cp -f ./p7_full_band_test_20m.bin /emulator
cp -f ./p8_full_band_test_20m.bin /emulator


#########################################################################
##  3. bin 파일출력
#########################################################################
cd /emulator
./emul_pcap_play p6_full_band_test_20m.bin


```

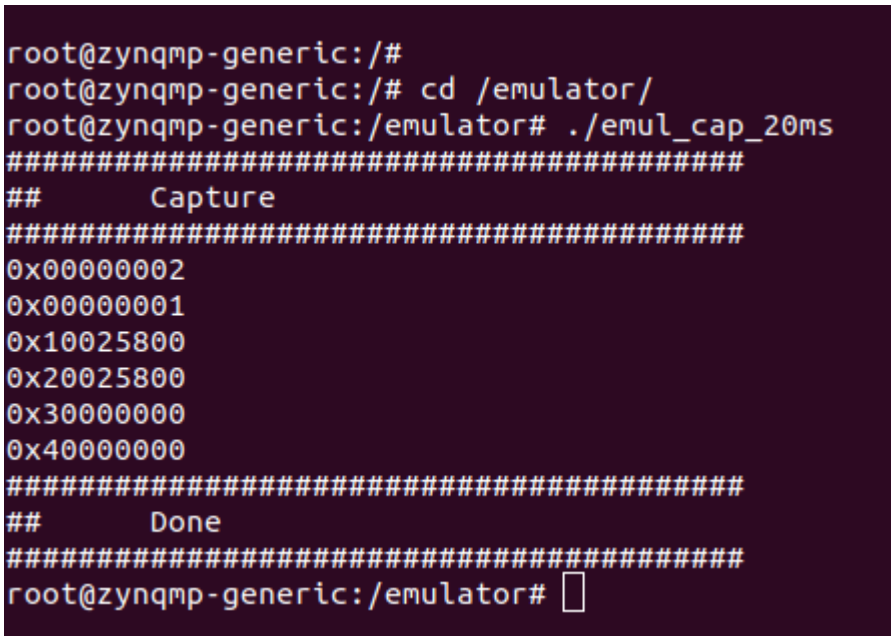
3. Capture Data & PCAP으로 저장

3.1 사용법-Capture

이 명령을 수행하면 수신한 Uplan 데이터를 메모리에 저장한다.

cd /emulator/

./emul_cap_20ms



3.2 저장된 패킷을 '.PCAP'으로 저장

메모리에 저장된 데이터를 아래 명령으로 PCAP파일로 저장한다.

cd /emulator/util

./save_port0 **<- Port0 저장**

./save_port1 **<- Port1 저장**

./save_port2 **<- Port2 저장**

./save_port3 **<- Port3 저장**

```
root@zynqmp-generic:/# cd /emulator/util/
root@zynqmp-generic:/emulator/util# ./save_port
save_port0  save_port1  save_port2  save_port3
root@zynqmp-generic:/emulator/util# ./save_port0
error / file exist (ultest.bin2k)
file(ultest.bin2k) branch(0) nframe(0)
show all controller info
  CTRL   : 0x(       0)
  STATUS: 0x(       1)
  BR(0) START : 0x(    1000) END   : 0x(10025800)
  BR(1) START : 0x(    2000) END   : 0x(20025800)
  BR(2) START : 0x(    3000) END   : 0x(30000000)
  BR(3) START : 0x(    4000) END   : 0x(40000000)
  ----------------
  LIST BR(0)
  PA64_CTRL  : 0x(        80a20000)
  PA64_START : 0x(       410000000)
  PA64_END   : 0x(       410025800)
  list       : (9600)
  ----------------

 auto size is (9600)
 final nframe :(9600)
 N( 0) offset(266c6000) blk64(  1536) ==> PA64(       5266c6000)
 N( 1) offset(266c6800) blk64(  1152) ==> PA64(       5266c6800)
 N( 2) offset(266c7000) blk64(   128) ==> PA64(       5266c7000)
 N( 3) offset(266c7800) blk64(   128) ==> PA64(       5266c7800)
 N( 4) offset(266c8000) blk64(   128) ==> PA64(       5266c8000)
 N( 5) offset(266c8800) blk64(   128) ==> PA64(       5266c8800)
 N( 6) offset(266c9000) blk64(   128) ==> PA64(       5266c9000)
```

중략.....

```
 n(8) pa64( 5266ca000) ts(               0 :  0.000000000) ncopy(   128)
 n(9) pa64( 5266ca800) ts(               0 :  0.000000000) ncopy(   128)
 n(10) pa64( 5266cb000) ts(              0 :  0.000000000) ncopy(  1536)
 n(11) pa64( 5266cb800) ts(              0 :  0.000000000) ncopy(  1152)
 n(12) pa64( 5266cc000) ts(              0 :  0.000000000) ncopy(  1536)
 n(13) pa64( 5266cc800) ts(              0 :  0.000000000) ncopy(  1152)
 n(14) pa64( 5266cd000) ts(              0 :  0.000000000) ncopy(  1536)
 n(15) pa64( 5266cd800) ts(              0 :  0.000000000) ncopy(  1152)
 n(16) pa64( 5266ce000) ts(              0 :  0.000000000) ncopy(  1536)
 n(17) pa64( 5266ce800) ts(              0 :  0.000000000) ncopy(  1152)
 n(18) pa64( 5266cf000) ts(              0 :  0.000000000) ncopy(  1536)
 n(19) pa64( 5266cf800) ts(              0 :  0.000000000) ncopy(  1152)
done
local pcapng
main
branch1 ./ultest.bin2k ./ultest_102_br0.pcapng 0 9600
---------------------
Opening output file
PKT_SIZE+FCS 1 1478
PKT_SIZE+FCS 2 1090
PKT_SIZE+FCS 3 1478
PKT_SIZE+FCS 4 1090
PKT_SIZE+FCS 5 1478
PKT_SIZE+FCS 6 1090
PKT_SIZE+FCS 7 1478
PKT_SIZE+FCS 8 1090
PKT_SIZE+FCS 9 1478
PKT_SIZE+FCS 10 1090
stop / file end
---------------------
total packet # =  9601
error packet # =  0
root@zynqmp-generic:/emulator/util#
```

4. Terminal 명령어

4.1 MAC 변경

PCAP 파일 재생하여 출력할 때, DST, SRC MAC주소를 변경하는 기능

4.1.1 MAC 변경 활성

##mac modify enable
devmem 0x80000020 32 0xff

##src-mac 66:55:44:33:22:11
devmem 0x80000070 6 0x0000665544332211

##R7 dst-mac a7:22:33:44:55:66
##R6 dst-mac a6:22:33:44:55:66
##R5 dst-mac a5:22:33:44:55:66
##R4 dst-mac a4:22:33:44:55:66
##R3 dst-mac a3:22:33:44:55:66
##R2 dst-mac a2:22:33:44:55:66
##R1 dst-mac a1:22:33:44:55:66
##R0 dst-mac a0:22:33:44:55:66
devmem 0x80000030 64 0x00006655443322a7
devmem 0x80000038 64 0x00006655443322a6
devmem 0x80000040 64 0x00006655443322a5
devmem 0x80000048 64 0x00006655443322a4
devmem 0x80000050 64 0x00006655443322a3
devmem 0x80000058 64 0x00006655443322a2
devmem 0x80000060 64 0x00006655443322a1
devmem 0x80000068 64 0x00006655443322a0

4.1.1 MAC 변경 비활성

devmem 0x80000020 32 0x00

4.2 Ping

Loopback으로 케이블을 연결하면 ping이 안됨

ifconfig eth1 xxx.xxx.xxx.xxx(Port IP)

ping xxx.xxx.xxx.xxx(상대IP)

```
#########################################################################
'Mac Table Reset'이 필요한 경우..... 노랑박스를 실행한다.
#########################################################################
#args 1 : Aging Time(Aging Time = time + time/2)
#args 2 : Aging Enable
#args 3 : Entry Clear
#hub_mactable(200, 1, 1)

devmem 0x80900008 32   (1<<25 | 1<<24 | 200)
sleep(2)
devmem 0x80900008 32   (0<<25 | 1<<24 | 200)
```

4.3 Fan Control

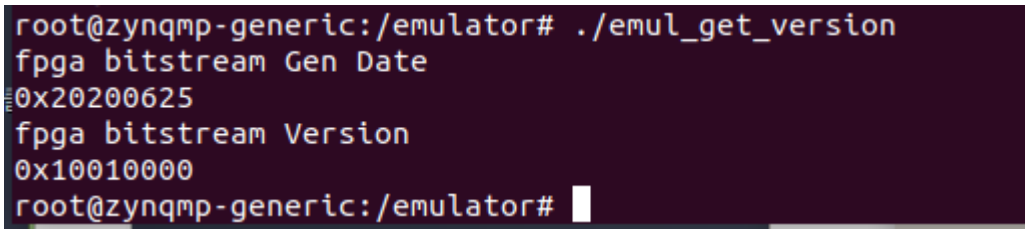
```
echo 10 > /sys/class/hwmon/hwmon1/pwm1
echo 10 > /sys/class/hwmon/hwmon1/pwm2
echo 10 > /sys/class/hwmon/hwmon1/pwm3
echo 10 > /sys/class/hwmon/hwmon1/pwm4
```

최대 RPM

```
echo 250 > /sys/class/hwmon/hwmon1/pwm1
echo 250 > /sys/class/hwmon/hwmon1/pwm2
echo 250 > /sys/class/hwmon/hwmon1/pwm3
echo 250 > /sys/class/hwmon/hwmon1/pwm4
```

4.4 emul_get_version

Fpga bitstream의 생성날짜와 버전을 알려준다.



생성날짜 : 2020.06.25

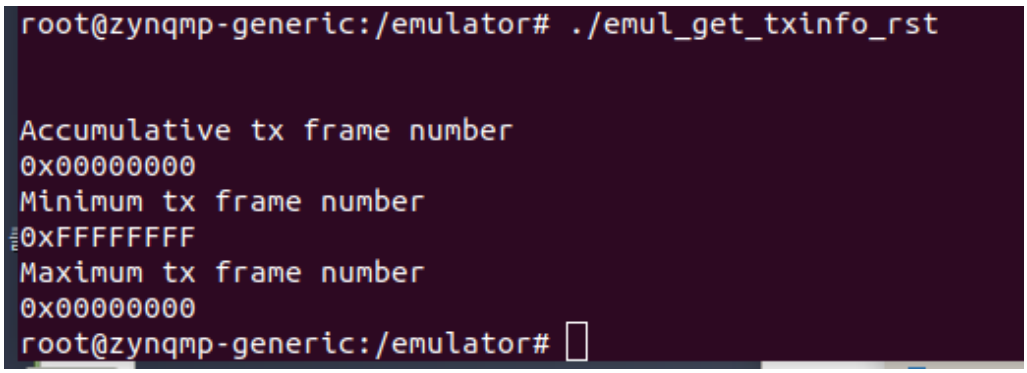
100_10000

type: 100(계측기)

Version: 10000

4.5 emul_get_txinfo_rst

TX Packet에 txframe 정보를 리셋한다.



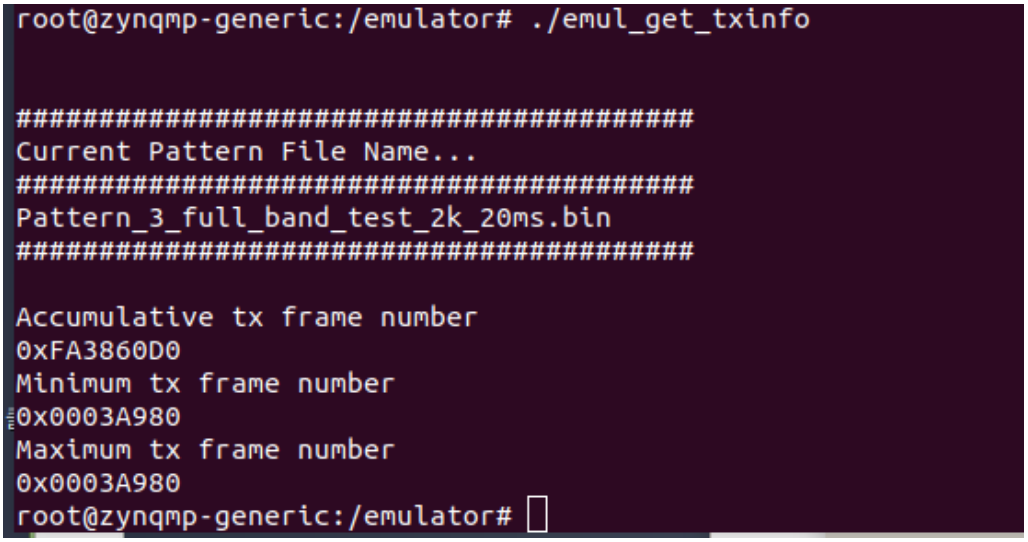
누적 프레임수는 0x0000_0000

단위 시간당 전송된 최소 프레임 수는 0xFFFF_FFFF

단위 시간당 전송된 최대 프레임 수는 0x0000_0000

4.6 emul_get_txinfo

단위 초당(1초) TX프레임 수와 실행되는 패턴파일 이름을 표시한다.
단위 시간당 전송되는 max, min값은 같다.



현재 재생되는 파일 이름이 표시된다. 'Pattern_3_full_band_test_2k_20ms.bin'

누적 프레임수는 '0xFA38_60D0'

단위 시간당 전송된 최소 프레임 수는 0x0003_A980

단위 시간당 전송된 최대 프레임 수는 0x0003_A980

리셋 후에 두 값은 항상 같아야 한다.

4.7 emul_get_rxinfo_rst

TBD

4.8 emul_get_rxinfo

TBD

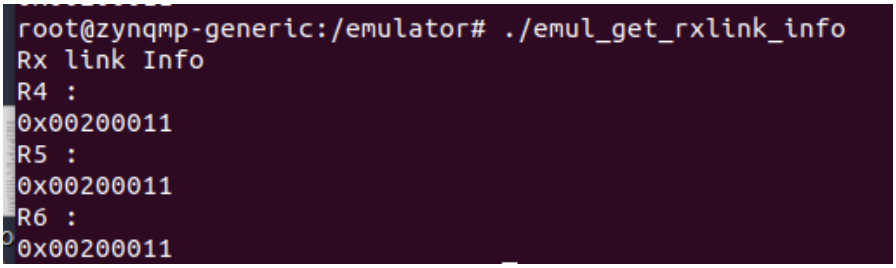
4.9 emul_get_rxlink_rst_info

FPGA와 POE 사이에 링크상태 레지스터를 리셋하고, 보여준다.

4.10 emul_get_rxlink_info

리셋 이후에 기록된 FPGA와 POE 사이에 링크상태를 보여준다.

다음과 같이 읽혀지는 것이 정상 상태임.



21	Rxlink-up
20	stat_rx_received_local_fault
19	stat_rx_internal_local_fault
18	stat_rx_remote_fault
17	stat_rx_local_fault
16	stat_rx_bad_sfd
15	stat_rx_bad_preamble
14	stat_rx_stomped_fcs
13	stat_rx_packet_bad_fcs
12	stat_rx_bad_fcs
11	stat_rx_toolong
10	stat_rx_undersize
9	stat_rx_oversize
8	stat_rx_packet_large
7	stat_rx_jabber
6	stat_rx_packet_small
5	stat_rx_bad_code
4	stat_rx_valid_ctrl_code
3	stat_rx_hi_ber
2	stat_rx_framing_err
1	stat_rx_framing_err_valid
0	Always '1'

5. LUA Script 명령

5.1. 패킷 카운트 모니터

R4: hubgetpc(0x48)
R5: hubgetpc(0x49)
R6: hubgetpc(0x4A)
L1 Optic: hubgetpc(0x83)

hubgetpc(0x48)

ex) # cd /emulator/util/
./Api

```
pleth-lua> hubgetpc(0x48)

########################## PROT[48] PRINT PACKET COUNTER ################################
    Egress(TX) Pkt Cnt
    ---> OctetCnt = 11527415808, UniCastCnt = 0, BroadCastCnt = 0, MultiCastCnt = 0
    ---> FixRout = 0,  PktCnt = 9587008, UCplanCnt = 9586992, MplanCnt = 0
    InGress(RX) Pkt Cnt
    ---> OctetCnt = 11527477920, UniCastCnt = 0, BroadCastCnt = 0, MultiCastCnt = 9587048
    ---> FixRout = 0,  PktCnt = 9587064, UCplanCnt = 9587048, MplanCnt = 0
```

U-Plan Data TX/RX 수를 보여준다.

5.2. POE 파워상태모니터

//args 1 : channel-mask(1:Enable, 0:Disable, lsb:channel0)

//args 2 : print time : 횟수

//args 3 : print interval 초

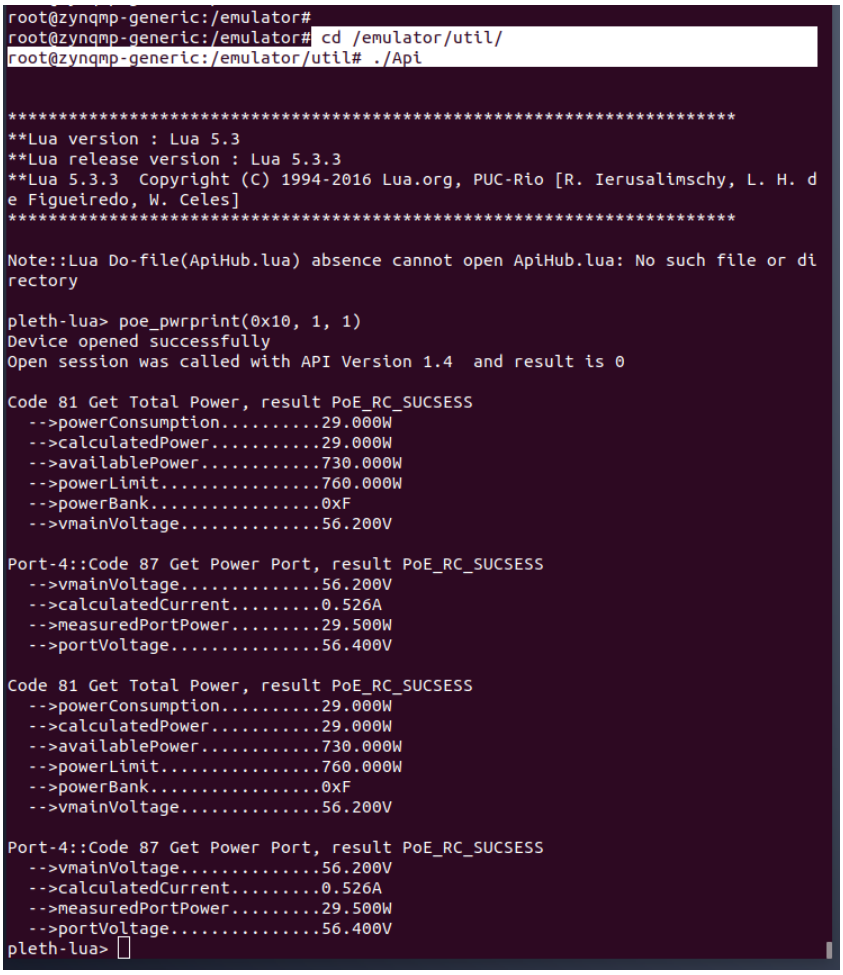
POE-R4: poe_pwrprint(0x10, 1, 1)

POE-R5: poe_pwrprint(0x20, 1, 1)

POE-R6: poe_pwrprint(0x40, 1, 1)

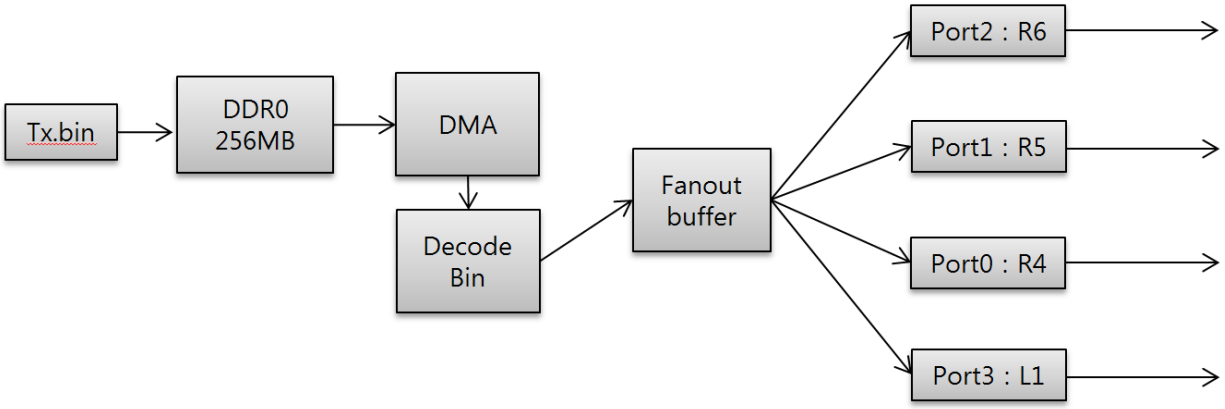
ex) # cd /emulator/util/

./Api



6. 사양 & 특성

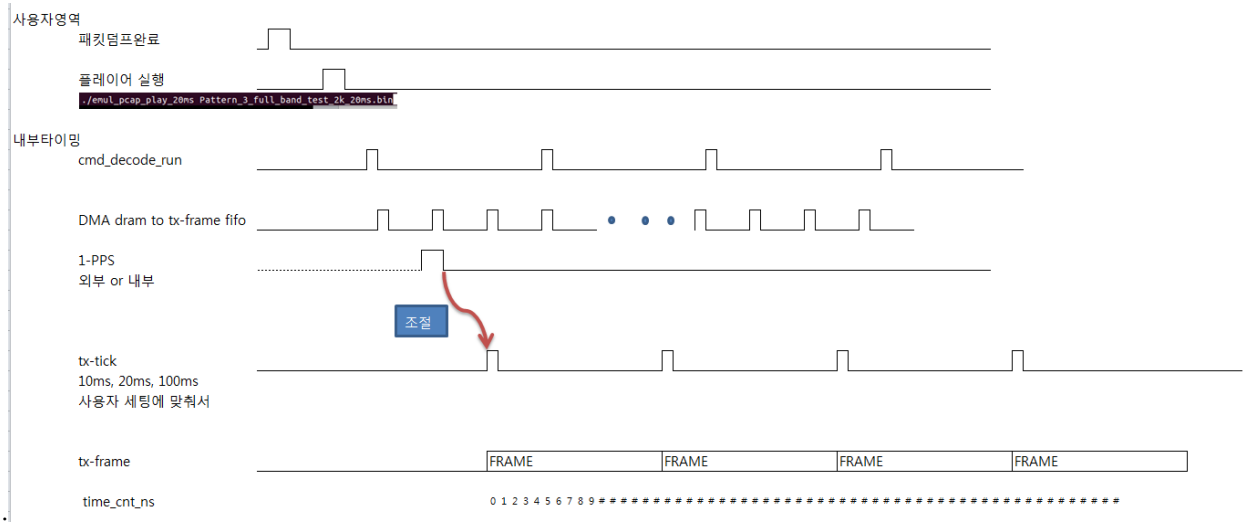
5.1 Player(TX)



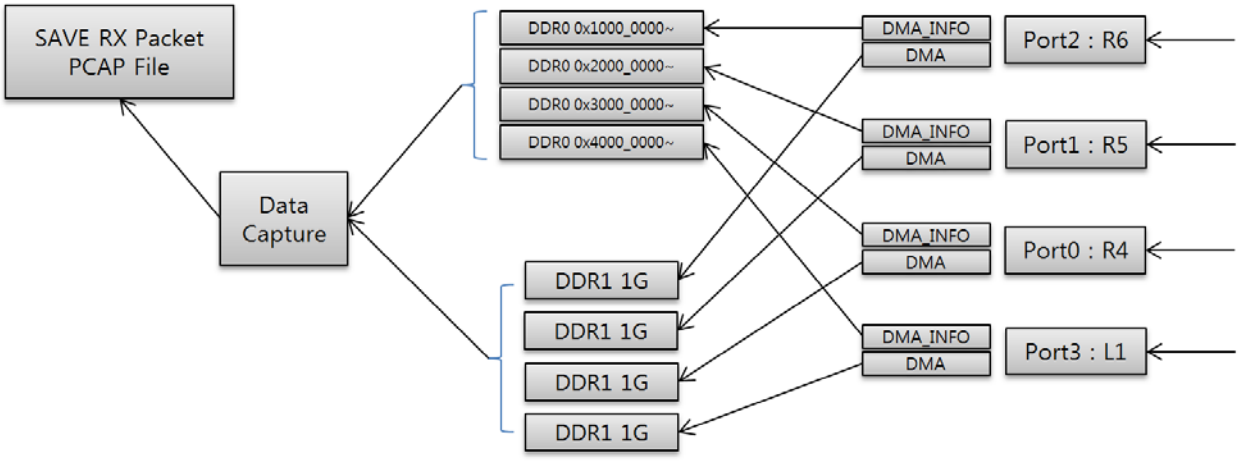
.bin 파일을 메모리 올려 놓고, 사용자가 설정한 10ms? 20ms? 주기로 읽어서 출력한다.

pcap 파일이 20ms 데이터 세트라면, 사용자는 20ms 이상으로 tx-tick을 설정해야 한다.

5.2 TX Packet Timing



5.3 Capture(RX)

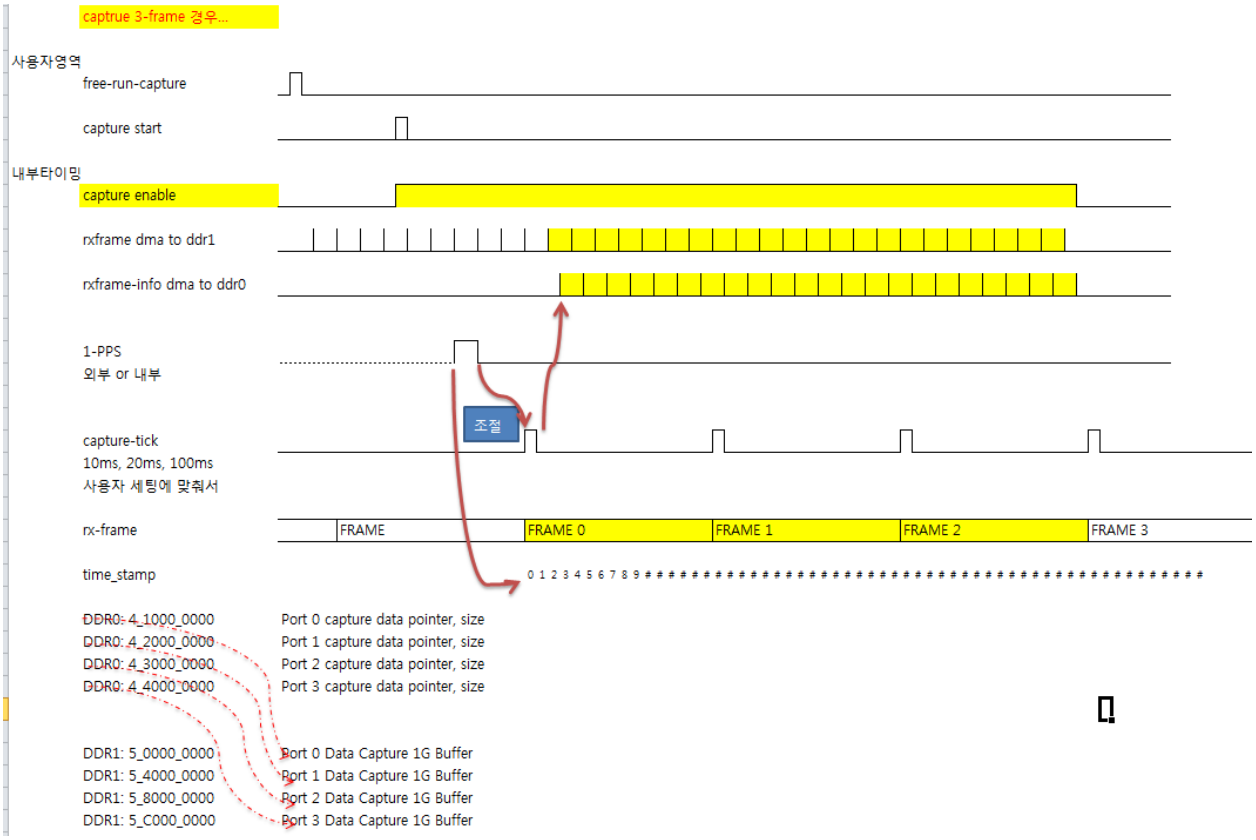


Capture로직은 U-Plane Data만 캡쳐가 가능하다. 최대 용량은 Port당 1GB만큼 저장이 가능하다.

'cap_re-run==1'이 되면 'DDR1'으로 패킷이 계속 저장된다. 'cap_run==1'이 되면, 1초 시스템 동기 신호에 cap_tick을 동기 시키고, 바로 이어오는 cap_tick에 맞춰서, 캡처를 시작한다. 캡처가 시작되면, DDR1에는 패킷이 저장되고, DDR0에는 DDR1에 저장되는 패킷의 저장주소가 저장된다. 캡처는 사용자가 설정한 cap_tick의 개수만큼 저장된다.

저장이 완료되면, DDR0에 저장된 DDR1의 주소를 참조해서 .PCAP 파일로 저장한다.

5.4 RX Packet Timing



7. Register Map

7.1MAC Control Register

MAC & Control

address	register	bit	type	Description
0x80000020	MAC overriding ENABLE	[7:0]	R/W	0xff: 재생하는 원본 pcap에 MAC을 변경하여 재생 0x00: 재생하는 원본 pcap 파일에 MAC을 그대로 재생
0x80000024	DlyAlign_En HUB_EmulatorMode HUB_DlyMode_output sel_1sec_pulse	[0] [8] [9] [13:12]	R/W R/W R/W R/W	항상 '1' 항상 '1' 항상 '0' 1초를 펄스만드는 소스를 선택한다. 이 1초 펄스는 내부 로직을 동기시키는 신호이다. 내부 로직은 pcap-플레이어, pcap-캡처로직이다. 이 동기 신호에 맞춰서 pcap을 재생하고, 캡처한다. 0: 시스템클락 156.25Mz를 소스로 1초펄스를 만들어 동기 신호로 사용한다 1. IDT Clock 소스에서 나오는 1초펄스를 사용한다 2: 외부 EXT 1PPS를 1초펄스로 사용한다.
0x80000028	CFG_PCID_Cmd CFG_PCID_CmdIdx CFG_PCID_Mu CFG_PCID_PktPerSym CFG_PCID_Id	[1:0] [5:2] [10:6] [15:12] [31:16]	R/W R/W R/W R/W R/W	
0x8000002C	CFG_PCID_Timeout CFG_PCID_OperFlag	[23:0] [31:24]	R/W R/W	
0x80000030	R0 destination MAC[63:0]		R/W	
0x80000038	R1 destination MAC[63:0]		R/W	
0x80000040	R2 destination MAC[63:0]		R/W	
0x80000048	R3 destination MAC[63:0]		R/W	
0x80000050	R4 destination MAC	[63:0]	R/W	MAC_Overriding_Enable == 1 인 경우 이 값으로 나간다.
0x80000058	R5 destination MAC	[63:0]	R/W	MAC_Overriding_Enable == 1 인 경우 이 값으로 나간다.
0x80000060	R6 destination MAC	[63:0]	R/W	MAC_Overriding_Enable == 1 인 경우 이 값으로 나간다.
0x80000068	CC destination MAC	[63:0]	R/W	MAC_Overriding_Enable == 1 인 경우 이 값으로 나간다.
0x80000070	source MAC	[63:0]	R/W	MAC_Overriding_Enable == 1 인 경우 이 값으로 나간다.

7.2 Pcap_player Register

Pcap_player

address	register	bit	type	Description
0x80A00000	reg_vector_address	[26:0]	R/W	tx.bin' 파일이 올라가는 메모리 주소 0x5_0000_0000 >> 9 = 항상 '0x02800000'
0x80A00004	reg_vector_length	[31:0]	R/W	패턴파일 ' X / 4096 +1' .... 값이 들어간다. 패턴파일 크기가 245472KB인 경우, 245472KB / 4096KB + 1 = **'60'**
0x80A00008	play_sts_err_cmdvector_seq play_sts_err_unknown play_sts_sts_ing play_sts_sts_cpl	[31] [30] [1] [0]	RO RO RO RO	tx.bin 파일에 문제가 있는 경우, 플레이어 재생중 예측되지 않은 문제가 있는 경우, 플레이어 재생중 플레이어가 'reg_player_lp_cnt' 만큼 재생되면 올라온다.
0x80A0000C	pcap_player_reset pcap_player_statistic_reset pcap_plaer_run	[31] [30] [0]	R/W R/W R/W	pcap 파일을 재생하기 전에 리셋이 필요함 pcap 플레이어에 tx-frame-counte를 리셋한다. pcap 파일 재생을 시작한다.
0x80A00010	reg_player_lp_cnt	[31:0]	R/W	패턴파일이 재생되는 횟수. 0: 무한반복 1: 1 회 2:2 회.....
0x80A00014	reg_tx_tick_gen_cnt	[31:0]	R/W	tx_tick의 주기를 설정한다. 기본값 (156,250,000-1) / 100 = 10ms 1,562,500 : 10ms 3,124,999 : 20ms
0x80A00024	tx_tick_delaycnt	[31:0]	R/W	1sec_pulse 기준으로 tx_tick이 발생하는 지연을 줄수 있다. pcap 플레이어는 tx_tick에 맞추어서 재생한다. 6.4ns/1-clk

7.3 Pcap_Capture Register

Pcap_capture

address	register	bit	type	Description
0x80A20000	R4 reg_cap_start_addr	[31:0]	R/W	항상 '0x1000' 캡처데이타가 저장된, 메모리 시작 주소
0x80A20004	R5 reg_cap_start_addr	[31:0]	R/W	항상 '0x2000' 캡처데이타가 저장된, 메모리 시작 주소
0x80A20008	R6 reg_cap_start_addr	[31:0]	R/W	항상 '0x3000' 캡처데이타가 저장된, 메모리 시작 주소
0x80A2000C	CC reg_cap_start_addr	[31:0]	R/W	항상 '0x4000' 캡처데이타가 저장된, 메모리 시작 주소
0x80A20010	reg_delayblock_reset reg_cap_rst reg_cap_prerun reg_cap_start	[31] [30] [1] [0]	R/W R/W R/W R/W	delay block reset... capture에 상위 로직 리셋 capture 로직 리셋 dram에 패킷을 저장을 시작한다. Start에 앞서 '1'을 준다 start가 '1'이 되면, capture_tick에 맞춰서 캡처 주소를 저장한다. reg_capframe_count 만큼 캡처를 하고 정지한다.
0x80A20014	capture_cpl capture_ing capture_err	[0] [1] [2]	RO RO RO	캡처 완료 캡처 진행중 캡처 에러
0x80A20018	R4 reg_cap_end_addr	[31:0]	RO	캡처데이타가 저장된, 메모리 마지막 주소
0x80A2001C	R5 reg_cap_end_addr	[31:0]	RO	캡처데이타가 저장된, 메모리 마지막 주소
0x80A20020	R6 reg_cap_end_addr	[31:0]	RO	캡처데이타가 저장된, 메모리 마지막 주소
0x80A20024	CC reg_cap_end_addr	[31:0]	RO	캡처데이타가 저장된, 메모리 마지막 주소
0x80A20028	reg_cap_tick_gen_cnt	[31:0]	R/W	capture_tick의 주기를 설정한다. 기본값 (156,250,000-1) / 100 = 10ms 1,562,500 : 10ms **3,124,999** : 20ms
0x80A2002C	reg_capframe_delay	[31:0]	R/W	1sec_pulse 기준으로 capture_tick이 발생하는 지연을 줄수 있다. capture는 capture_tick에 맞추어서 저장한다. 6.4ns/1-clk
0x80A20030	reg_capframe_count	[31:0]	R/W	capture frame 수를 설정한다. 현재는 1~4 Frame 저장가능

8..... TBD