Daisy Device Test Guide

Revision 1.0 2019. 11. 27

CRZ Technology http://www.mangoboard.com/

Document History

Revision	Date	Change note
1.0	2019.11.27	Initial Version

목 차

1.	개요		4
2.	DIMM slo	ot 테스트	5
3.	PCle x16	Endpoint 테스트	7
	3.1.	PCIe Host BIOS 설정	7
	3.2.	PCle 링크 검증하기	7
	3.3.	XDMA 테스트	13
4.	QSFP28 2	채널 테스트	. 15
	4.1.	Passive optical loopback card	15
	4.2.	Passive 1m copper cable	18
	4.3.	Passive 2m copper cable	19
	4.4.	Active optical cable	21
5.	M.2 테스	<u>E</u>	. 24

1. 개요

이 문서는 Daisy 보드에 장착되어 있는 디바이스들(DIMM2개, PCle x16 Endpoint, QSFP28 2개, NVMe M.2 2개)의 테스트 절차를 설명한다.

2. DIMM slot 테스트

테스트 진행시 Mode Select DIP Switch[SW1]을 JTAG 모드로 설정한다.





MODE		Swi	tch	
MODE	[4]	[3]	[2]	[1]
JTAG	LOW	LOW	LOW	LOW
QSPI 32	LOW	LOW	HIGH	LOW
SD1	HIGH	HIGH	HIGH	LOW

테스트에 이용한 RDIMM은 J8에 장착한 8GB x8 DDR4_M393A1K43BB0-CRC과 J9에 장착한 16GB x4 DDR4_M393A2K40BB1-CRC 이다.

USB 케이블을 PC와 연결하고 12V 전원을 연결한 다음 전원 스위치를 켠다.

제공된 Vivado 프로젝트(<u>dual_mig_x8.zip</u>)를 오픈한다. 사용된 Vivado 버전은 2019.1 이다.

Address Editor에서 8GB RDIMM은 0x100000000에, 16GB RDIMM은 0x1800000000에 매핑되어 있는 것을 알 수 있다.

Diagram 🗙 Address Editor 🗙	c .				
Q ≚ ≑ ⊯					
Cell	Slave Interface	Base Name	Offset Address	Range	High Address
∨ ‡ zynq_ultra_ps_e_0					
✓ III Data (40 address bits : 0x00	040000000 [256M] ,0x040	0000000 [4G] ,0x1000000000 [22	4G])		
ddr4_0	C0_DDR4_S_AXI	C0_DDR4_ADDRESS_BLOCK	0×10_0000_0000	8G 👻	0x11_FFFF_FFFF
ddr4_0	C0_DDR4_S_AXI_CTRL	C0_REG	0×00_A000_0000	4K 👻	0x00_A000_0FFF
ddr4_1	C0_DDR4_S_AXI	C0_DDR4_ADDRESS_BLOCK	0×18_0000_0000	16G 👻	0x1B_FFFF_FFFF
ddr4_1	C0_DDR4_S_AXI_CTRL	C0_REG	0×00_A000_1000	4K •	0x00_A000_1FFF

"PROGRAM AND DEBUG" 아래 "Open Hardware Manager"를 선택하여 "Program device"를 클릭 한다.

FPGA가 프로그램된 후 LED0, LED1이 켜지는지 확인한다. MIG Calibration이 정상이면 LED 2개가 켜진다.

디버그 메시지를 보기 위해서 터미널을 실행한다.

RDIMM 접근이 정상인지 보기 위해 "File" -> "Launch SDK"를 선택하여 SDK를 론칭한다. SDK 상에서 "Run" -> "Debug History" -> "System Debugger on Local"을 선택한다.

FPGA 프로그램이 끝나면 LED 2개가 켜지는지 확인한다.

Cortex-A53 #0를 클릭하고 F8을 눌러 실행한다. 메모리 테스트가 패스하는지 확인한다.

📕 COM10:115200baud - Tera Term VT	—	
<u>Eile Edit Setup Control Window H</u> elp		
Starting Memory Test Application		
NOTE: This application runs with D-Cache disabled.As a result, cache	line	request
s will not be generated		
Testing memory region: ddr4_0_C0_DDR4_ADDRESS_BLOCK		
Memory Controller: ddr4_0		
Base Address: 0x100000000		
Size: 0x20000000 bytes		
32-bit test: PASSED!		
16-bit test: PASSED!		
8-bit test: PASSED!		
Testing memory region: ddr4_1_C0_DDR4_ADDRESS_BLOCK		
Memory Controller: ddr4_1		
Base Address: 0x180000000		
Size: 0x400000000 bytes		
32-bit test: PASSED!		
16-bit test: PASSED!		
8-bit test: PASSED!		
Memory Test Application Complete		

3. PCIe x16 Endpoint 테스트

테스트 진행시 Mode Select DIP Switch[SW1]을 JTAG 모드로 설정한다.

3.1. PCIe Host BIOS 설정

PCle Host BIOS 설정에서 PCle 슬롯을 활성화시켜야 한다. 검증에 이용한 HOST는 HP PRODESK인데 BIOS에서 아래와 같이 PCle 슬롯을 활성화해야 한다.

	Main Security
+	System Options
	Configure Storage Controller for Inte
	Turbo-boost
	Hyperthreading 🚱
	Multi-processor
	Virtualization Technology (VTx)
	Virtualization Technology for Directed
	PCI Express x16 Slot 1 🚱
	PCI Express x1 Slot 1 🚱
	PCI Express x4 Slot 1 🚱
	M.2 WLAN/BT
	Allow PCIe/PCI SERR# Interrupt
	Power Button Override

3.2. PCle 링크 검증하기

Daisy 보드에 PCle extension cable을 장착한 다음 PRODESK의 PCle x16 슬롯에 꽂는다.(PCle 슬롯 3개 중 맨 위의 검은 색 슬롯이다.) 보드에 12V 전원을 인가한다. 제공된 Vivado 프로젝트(<u>pcie_ep.zip</u>)를 오픈하여 bitstream을 다운로드한다. PRODESK의 전원을 켠 다음 리눅스에 로그인한다. 이때 보드의 LED0이 점등하는지 확인한다.

```
리눅스에 로그인한 다음 command line에서 아래 명령을 수행하여 링크가 설정되었는지 확인한다.

xilinx@xilinx-HP-ProDesk-400-G4-MT:~

File Edit View Search Terminal Help

xilinx@xilinx-HP-ProDesk-400-G4-MT:~$ lspci -vvv -s 01:00.0

01:00.0 Serial controller: Xilinx Corporation Device 903f (prog-if 01 [16450])

Subsystem: Xilinx Corporation Device 0007

Control: I/O- Mem+ BusMaster- SpecCycle- MemWINV- VGASnoop- ParErr- Step

ping- SERR+ FastB2B- DisINTx-

Status: Cap+ 66MHz- UDF- FastB2B- ParErr- DEVSEL=fast >TAbort- <TAbort-

<MAbort- >SERR- <PERR- INTx-

Interrupt: pin A routed to IRQ 16

Region 0: Memory at f0100000 (32-bit, non-prefetchable) [size=64K]

Capabilities: <access denied>

xilinx@xilinx-HP-ProDesk-400-G4-MT:~$
```

다음은 vivado 툴 상에서 PCIe 링크를 검증하는 방법이다. 보드와 JTAG이 연결된 상태에서 실행 해야 한다.

Tcl Console에서 pcie_ep.ip_user_files/mem_init_files 디렉토리로 이동한다.



test_rd.tcl을 실행한다.

Tcl Console 🗙 Messages	Log Reports Design Runs
Q ¥ ♦ 🗉 目	
2019-09-23 오후 04:04	96 pcie_debug_rst_trc.dat
2019-09-23 오후 04:04	96 pcie_debug_static_info.dat
2019-09-23 오후 04:04	768 rxdet.dat
2019-09-10 오후 05:30	4,594 test_rd.tcl
9개 파일	34,190 비이트
🚊 2개 디렉터리	448,869,003,264 바이트 남음
<	
source ./test_rd.tcl	

pcie_debug_info_trc.dat / pcie_debug_ltssm_trc.dat / pcie_debug_rst_trc.dat /

pcie_debug_static_info.dat / rxdet.dat 등 다섯개의 dat 파일이 새로 생성된 것을 확인할 수 있다.

📙 🛃 📕 🖛 me	m_init	_files			
파일 홈 공위	R	보기			
$\leftarrow \rightarrow \cdot \cdot \uparrow$	« Xi	inx > Daisy > PCle > EP > pcie_ep >	pcie_ep.ip_user_files > mem_ini	t_files	ٽ ~
🛃 바로 가기		이름	수정한 날짜	유형	크기
및 비타 치며		🐼 draw_ltssm.tcl	2019-09-10 오후 5:30	TCL 파일	12KB
- 이상 외전	7	🐼 draw_reset.tcl	2019-09-10 오후 5:30	TCL 파일	6KB
➡ 나운도느	Ŕ	🐼 draw_rxdet.tcl	2019-09-10 오후 5:30	TCL 파일	6KB
🔮 문서	*	pcie_debug_info_trc.dat	2019-11-27 오후 2:20	DAT 파일	1KB
📰 사진	1	pcie_debug_ltssm_trc.dat	2019-11-27 오후 2:20	DAT 파일	6KB
dual_mig_x8		pcie_debug_rst_trc.dat	2019-11-27 오후 2:20	DAT 파일	1KB
latest		pcie_debug_static_info.dat	2019-11-27 오후 2:20	DAT 파일	1KB
mem init files		📄 rxdet.dat	2019-11-27 오후 2:20	DAT 파일	1 KB
Xilinx	,	🛞 test_rd.tcl	2019-09-10 오후 5:30	TCL 파일	5KB

Daisy Device Test Guide

Windows explorer에서 draw_reset.tcl / draw_rxdet.tcl / draw_ltssm.tcl를 더블클릭하여 정상적인지 확인한다.

 <u>.</u>

draw_reset.tcl - PHY 리셋시 정상동작함을 확인할 수 있다.

draw_rxdet.tcl - 16개 Lane이 전부 이상 없음을 알 수 있다.





draw_ltssm.tcl – link training status와 state machine도 정상적이다.

Serial I/O Links 탭에서 Create Scan...을 선택한다.

Tcl Console Messag	es Serial I/O Link	s _× Serial I	O Scans							
Q ¥ € ╋										
Name C	Create Links	RX	TX Pre-Cursor		TX Post-Cursor		TX Diff Swing	DFE Enabled		
😑 Ungrouped Lin	Create Link Group									
√ ⊗ Link Group 0 (C C C C C C C C C C C C C C C C C C C	Create Scan		User Value	~	User Value	~	User Value 🗸 🗸	User Value	\sim	
S Link 0	Create Sween	MGT_X0Y0/RX	User Value	~	User Value	\sim	User Value 🗸 🗸	User Value	\sim	
🗞 Link 1 🗌		MGT_X0Y1/RX	User Value	~	User Value	\sim	User Value v	User Value	~	
N Link 10	MGT_X0Y10/TX	MGT_X0Y10/RX	User Value	~	User Value	~	User Value 🗸 🗸	User Value	~	
N Link 11	MGT_X0Y11/TX	MGT_X0Y11/RX	User Value	~	User Value	\sim	User Value 🗸 🗸	User Value	\sim	
N Link 12	MGT_X0Y12/TX	MGT_X0Y12/RX	User Value	~	User Value	~	User Value v	User Value	~	
N Link 13	MGT_X0Y13/TX	MGT_X0Y13/RX	User Value	~	User Value	~	User Value 🗸 🗸	User Value	~	
N Link 14	MGT_X0Y14/TX	MGT_X0Y14/RX	User Value	~	User Value	\sim	User Value 🗸	User Value	\sim	
N Link 15	MGT_X0Y15/TX	MGT_X0Y15/RX	User Value	~	User Value	~	User Value 🗸 🗸	User Value	~	

Daisy Device Test Guide

Link0부터 Link15까지 추가하여 eye diagram이 이상이 없는지 확인한다.

🍌 Create Scan			×
Set the description a on the selected link.	nd oth	er properties to create and optionally run a scan	4
Link: Link) (MGT	_X0Y0/TX, MGT_X0Y0/RX)	~
Description: Scan	0		\otimes
Scan Properties			
<u>S</u> can type:		2D Full Eyescan	-
<u>H</u> orizontal incre	ment	8	-
Horizontal range	E.	-0.500 UI to 0.500 UI	-
<u>V</u> ertical increme	nt	8	-
V <u>e</u> rtical range:		100%	-
Dwell			
) <u>B</u> ER: 1e-5	5	•	-
○ <u>T</u> ime:		0	*
✓ <u>R</u> un scan			
?		OK	cel



3.3. XDMA 테스트

다음 링크에 접속하여 필요한 파일을 Ubuntu 16.04에 복사한다. <u>https://github.com/Xilinx/dma_ip_drivers</u>

다음 명령을 실행하여 XDMA 드라이버 커널 모듈과 응용 프로그램을 빌드한다.

\$ cd XDMA/linux-kernel

\$ cd xdma

\$ make install

\$ cd tools

\$ make

\$ cd tests

XDMA 커널 모듈을 로딩한다. \$ sudo ./load_driver.sh

XDMA가 정상 동작하는지 테스트한다. \$./run_test.sh

```
xilinx@xilinx:~/Downloads/dma ip drivers-master/XDMA/linux-kernel/tests$ sudo ./
run test.sh
Info: Number of enabled h2c channels = 1
Info: Number of enabled c2h channels = 1
Info: The PCIe DMA core is memory mapped.
Info: Running PCIe DMA memory mapped write read test
      transfer size: 1024
      transfer count: 1
Info: Writing to h2c channel 0 at address offset 0.
Info: Wait for current transactions to complete.
** Average BW = 1024, 17.526144
Info: Writing to h2c channel 0 at address offset 1024.
Info: Wait for current transactions to complete.
** Average BW = 1024, 11.151161
Info: Writing to h2c channel 0 at address offset 2048.
Info: Wait for current transactions to complete.
** Average BW = 1024, 13.890962
Info: Writing to h2c channel 0 at address offset 3072.
Info: Wait for current transactions to complete.
** Average BW = 1024, 16.115833
Info: Reading from c2h channel 0 at address offset 0.
Info: Wait for the current transactions to complete.
** Average BW = 1024, 2.690623
Info: Reading from c2h channel 0 at address offset 1024.
Info: Wait for the current transactions to complete.
** Average BW = 1024, 4.612342
Info: Reading from c2h channel 0 at address offset 2048.
Info: Wait for the current transactions to complete.
** Average BW = 1024, 4.601089
Info: Reading from c2h channel 0 at address offset 3072.
Info: Wait for the current transactions to complete.
** Average BW = 1024, 4.605227
Info: Checking data integrity.
Info: Data check passed for address range 0 - 1024.
Info: Data check passed for address range 1024 - 2048.
Info: Data check passed for address range 2048 - 3072.
Info: Data check passed for address range 3072 - 4096.
Info: All PCIe DMA memory mapped tests passed.
Info: All tests in run_tests.sh passed.
xilinx@xilinx:~/Downloads/dma_ip_drivers-master/XDMA/linux-kernel/tests$
```

4. QSFP28 2채널 테스트

테스트 진행시 Mode Select DIP Switch[SW1]을 JTAG 모드로 설정한다.

테스트에 이용한 100G Ethernet card는 다음과 같다.

Passive opticable loopback card https://kr.mouser.com/ProductDetail/Amphenol-Commercial-Products/SF-100GLB35W-

0DB?qs=%2Fha2pyFaduhfF1djD4B3nSqHSlyoiiFwuFIAWTkNsVXI5eAQcyhMRNkUlzug2Tuz

Passive 1m copper cable

https://www.digikey.kr/product-detail/ko/te-connectivity-amp-connectors/2333393-3/A142599-ND/9922312

Passive 2m copper cable <u>https://www.digikey.kr/product-detail/ko/te-connectivity-amp-connectors/2333393-5/A142600-</u> <u>ND/9922313</u>

Active optical cable

4.1. Passive optical loopback card

루프백 카드를 J20의 QSFP28 #0에 삽입한다.

제공된 Vivado 프로젝트(cmac_usplus_0_ex_vio_1_2.zip)를 오픈하여 bitstream을 다운로드한다.

아래 그림과 같이 테스트가 실패하는 것을 알 수 있다.

hw_	_vios											
	hw_vio_1											
tions	Q ¥ ≑ + -											
d o p	Name	Value		Acti	Directi	VIO						
boar	Ъ rx_aligned_led	•			Input	hw_vio_1						
ash	□ rx_gt_locked_led	•			Input	hw_vio_1						
	□ rx_done_led	•			Input	hw_vio_1						
	l₀ tx_done_led	•			Input	hw_vio_1						
	l₀ rx_busy_led	۲			Input	hw_vio_1						
	□ ₀ tx_busy_led	۲			Input	hw_vio_1						
	Ъ_ rx_data_fail_led	0			Input	hw_vio_1						
	↓ Ibus_tx_rx_restart_in	0			Output	hw_vio_1						
	ີ asend_continuous_pkts	0			Output	hw_vio_1						
	l₀ sys_reset	0			Output	hw_vio_1						
	> 🖫 gt_loopback_in[11:0]	[H] 000			Output	hw_vio_1						
	> 🗓 gt_txdiffctrl[19:0]	[H] 0_0000			Output	hw_vio_1						
	> 🗓 gt_txpostcursor[19:0]	[H] 0_0000			Output	hw_vio_1						
	> 🗓 gt_txprecursor[19:0]	[H] 0_0000	Ŧ		Output	hw_vio_1						

이런 경우 제공된 Vivado 프로젝트(<u>ibert_ultrascale_gty_0_ex_25G.zip</u>)를 오픈하여 bitstream을 다 운로드한다.

아래 그림과 같이 TX Pattern – PRBS 31-bit, RX Pattern – PRBS 32-bit에 대해서 Errors가 0이 되는 TX Pre-Cursor, TX Post-Cursor, TX Diff Swing 값을 찾아서 cmac_usplus_0_ex_vio_1_2의 VIO에 설정 한다.

Tcl Console Messages	Serial I/O Lin	ks _× Serial	I/O Scans															
Name	тх	RX	Status	Bits	Errors	BER	BERT Reset	TX Pattern		RX Pattern		TX Pre-Cursor		TX Post-Cursor		TX Diff Swing		DFE Enabled
Ungrouped Links (0)																		
🗸 🛞 Link Group 0 (8)							Reset	PRBS 31-bit	~	PRBS 31-bit	~	1.16 dB (00101)	~	6.02 dB (10100)	~	470 mV (00011)	~	2
% Link 0	MGT_X0Y12/T)	MGT_X0Y12/RX	25.781 Gbps	1.499E11	0E0	6.669E-12	Reset	PRBS 31-bit	~	PRBS 31-bit	~	1.16 dB (00101)	~	6.02 dB (10100)	~	470 mV (00011)	\sim	1
% Link 1	MGT_X0Y13/T)	MGT_X0Y13/RX	25.781 Gbps	1.5E11	0E0	6.668E-12	Reset	PRBS 31-bit	~	PRBS 31-bit	~	1.16 dB (00101)	~	6.02 dB (10100)	~	470 mV (00011)	\sim	1
% Link 2	MGT_X0Y14/T)	MGT_X0Y14/RX	25.781 Gbps	1.5E11	0E0	6.667E-12	Reset	PRBS 31-bit	~	PRBS 31-bit	~	1.16 dB (00101)	~	6.02 dB (10100)	~	470 mV (00011)	\sim	1
N Link 3	MGT_X0Y15/T)	MGT_X0Y15/RX	25.779 Gbps	1.5E11	0E0	6.666E-12	Reset	PRBS 31-bit	~	PRBS 31-bit	~	1.16 dB (00101)	~	6.02 dB (10100)	~	470 mV (00011)	\sim	1
No Link 4	MGT_X0Y16/T)	MGT_X0Y16/RX	No Link	1.499E11	7.674E10	5.118E-1	Reset	PRBS 31-bit	~	PRBS 31-bit	~	1.16 dB (00101)	~	6.02 dB (10100)	~	470 mV (00011)	\sim	1
No Link 5	MGT_X0Y17/T)	MGT_X0Y17/RX	No Link	1.5E11	1.425E11	9.5E-1	Reset	PRBS 31-bit	~	PRBS 31-bit	~	1.16 dB (00101)	~	6.02 dB (10100)	~	470 mV (00011)	\sim	1
Na Link 6	MGT_X0Y18/T)	MGT_X0Y18/RX	No Link	1.501E11	7.337E10	4.887E-1	Reset	PRBS 31-bit	~	PRBS 31-bit	~	1.16 dB (00101)	~	6.02 dB (10100)	\sim	470 mV (00011)	~	v
No Link 7	MGT_X0Y19/T)	MGT_X0Y19/RX	No Link	1.499E11	1.424E11	9.5E-1	Reset	PRBS 31-bit	~	PRBS 31-bit	~	1.16 dB (00101)	~	6.02 dB (10100)	~	470 mV (00011)	~	2
í																		

hw_	vios					? 🗆 🖒 🗙
	hw_vio_1					? _ 🗆 X
tions	Q ¥ ♦ + -					
d O D	Name	Value	Activity	Direction	VIO	
boar	l_{\bullet} rx_aligned_led	•		Input	hw_vio_1	
ash	□ rx_gt_locked_led	0		Input	hw_vio_1	
	□ rx_done_led	•		Input	hw_vio_1	
	□ tx_done_led	•		Input	hw_vio_1	
	□ rx_busy_led	۲		Input	hw_vio_1	
	↓ tx_busy_led	۲		Input	hw_vio_1	
	□ rx_data_fail_led	۲		Input	hw_vio_1	
	□ Ibus_tx_rx_restart_in	0		Output	hw_vio_1	
	l_{\bullet} send_continuous_pkts	0		Output	hw_vio_1	
	l₄ sys_reset	0		Output	hw_vio_1	
	> 1 gt_loopback_in[11:0]	[H] 000	•	Output	hw_vio_1	
	> 1 gt_txprecursor[19:0]	[H] 2_94A5	•	Output	hw_vio_1	
	> 1 gt_txpostcursor[19:0]	[H] A_5294	•	Output	hw_vio_1	
	> 1 gt_txdiffctrl[19:0]	[H] 1_8C63	•	Output	hw_vio_1	

적절한 TX Pre-Cursor / TX Post-Cursor / TX Diff Swing을 설정하면 테스트가 통과함을 알 수 있다.

루프백 카드를 J23의 QSFP28 #1에 삽입한다.

아래 그림과 같이 테스트가 성공하는 것을 알 수 있다. 이 경우 IBERT를 이용하여 TX Pre-Cursor / TX Post-Cursor / TX Diff Swing을 찾지 않아도 된다.

das	hboard_1				
	hw_vio_2				
tions	Q ¥ € + -				
dop	Name	Value	Activity	Direction	VIO
boar	Ъ rx_aligned_led2	•		Input	hw_vio_2
Dash	Trx_gt_locked_led2	•		Input	hw_vio_2
	□ rx_done_led2	•		Input	hw_vio_2
	Ъ tx_done_led2	•		Input	hw_vio_2
	□ rx_busy_led2	•		Input	hw_vio_2
	ີ⊾ tx_busy_led2	•		Input	hw_vio_2
	□ rx_data_fail_led2	•		Input	hw_vio_2
	↓ Ibus_tx_rx_restart_in2	0		Output	hw_vio_2
	↓ send_continuous_pkts2	0		Output	hw_vio_2
	ી _∎ sys_reset2	0		Output	hw_vio_2
	> 1 gt_loopback_in2[11:0]	[H] 000 👻		Output	hw_vio_2
	> 🗓 gt_txdiffctrl2[19:0]	[H] 0_0000 🔹		Output	hw_vio_2
	> 🗓 gt_txpostcursor2[19:0]	[H] 0_0000 🔹		Output	hw_vio_2
	> 🗓 gt_txprecursor2[19:0]	[H] 0_0000 🔻		Output	hw_vio_2

4.2. Passive 1m copper cable

1m copper cable을 J20의 QSFP28 #0과 J23의 QSFP28 #1에 삽입한다.

아래 그림처럼 이 경우 둘 중 하나의 루프백 모드를 Far-End PMA loopback 모드로 설정하여야 한다.

hw	_vios					? 🗆 🗆 X	C	las	hboard_1					
	hw_vio_1					? _ 🗆 X	н.		hw_vio_2					
tions	Q ¥ ♦ + -						enot		Q ¥ ♦ + -					
do p	Name	Value	Activity	Direction	VIO		ć	5	Name	Value		Activity	Direction	VIO
boan	Tr_aligned_led	•		Input	hw_vio_1				7, rx_aligned_led2	•			Input	hw_vio_2
ash	l₀ rx_gt_locked_led	•		Input	hw_vio_1		4 ec	20	□ rx_gt_locked_led2	•			Input	hw_vio_2
	l₀ rx_done_led	•		Input	hw_vio_1			-	<pre>led2</pre>	•			Input	hw_vio_2
	l₀ tx_done_led	•		Input	hw_vio_1				□ tx_done_led2	•			Input	hw_vio_2
	l₀ rx_busy_led	•		Input	hw_vio_1				<pre>led2</pre>	۲			Input	hw_vio_2
	l₀ tx_busy_led	•		Input	hw_vio_1				∃e tx_busy_led2	٠			Input	hw_vio_2
	Trx_data_fail_led	•		Input	hw_vio_1				la rx_data_fail_led2	۲			Input	hw_vio_2
	la_lbus_tx_rx_restart_in	0		Output	hw_vio_1				lbus_tx_nx_restart_in2	0			Output	hw_vio_2
	∃ send_continuous_pkts	0		Output	hw_vio_1				□ send_continuous_pkts2	0			Output	hw_vio_2
	l₀ sys_reset	0]	Output	hw_vio_1				l₄ sys_reset2	0			Output	hw_vio_2
	> 1 gt_loopback_in[11:0]	[H] 924 💌		Output	hw_vio_1				> 1 gt_loopback_in2[11:0]	[H] 000	*		Output	hw_vio_2
	> 🗓 gt_txprecursor[19:0]	[H] 0_0000 🔹		Output	hw_vio_1				> 1 gt_txdiffctrl2[19:0]	[H] 0_0000	~		Output	hw_vio_2
	> "l_ gt_txpostcursor[19:0]	[H] 0_0000 🔹		Output	hw_vio_1				> 1 gt_txpostcursor2[19:0]	[H] 0_0000	*		Output	hw_vio_2
	> 🗓 gt_txdiffctrl[19:0]	[H] 0_0000 🔹		Output	hw_vio_1				> 🗓 gt_txprecursor2[19:0]	[H] 0_0000	~		Output	hw_vio_2

QSFP28 #0을 Far-End PMA loopback모드로 설정한 경우

QSFP28 #1을 Far-End PMA loopback모드로 설정한 경우

vios					? 🗆 🗆 ×	da	shboard_1				
hw_vio_1					? _ 🗆 ×		hw_vio_2				
Q ¥ ≑ + =						ions	Q ¥ ≑ + =				
Name	Value	Activity	Direction	VIO		dop	Name	Value	Activity	Direction	VIO
Tx_aligned_led	•		Input	hw_vio_1		poar	l₀ rx_aligned_led2	•		Input	hw_vio_2
Trx_gt_locked_led	•		Input	hw_vio_1		ldsh	lackstart	•		Input	hw_vio_2
T_ rx_done_led	•		Input	hw_vio_1			l₀ rx_done_led2	•		Input	hw_vio_2
↓ tx_done_led	•		Input	hw_vio_1			l₀ tx_done_led2	•		Input	hw_vio_2
<pre>led</pre>	•		Input	hw_vio_1			led2	٠		Input	hw_vio_2
l₀ tx_busy_led	•		Input	hw_vio_1			l₀ tx_busy_led2	•		Input	hw_vio_2
T_ rx_data_fail_led	•		Input	hw_vio_1			l_ rx_data_fail_led2	۰		Input	hw_vio_2
↓ Ibus_tx_n_restart_in	0		Output	hw_vio_1			la_lbus_tx_rx_restart_in2	0		Output	hw_vio_2
□ send_continuous_pkts	0		Output	hw_vio_1			l _e send_continuous_pkts2	0		Output	hw_vio_2
l₄ sys_reset	0		Output	hw_vio_1			l₀ sys_reset2	0		Output	hw_vio_2
> 1, gt_loopback_in[11:0]	[H] 000 👻		Output	hw_vio_1			> 1 gt_loopback_in2[11:0]	[H] 924	•	Output	hw_vio_2
> 1, gt_txprecursor[19:0]	[H] 0_0000 🔻	·	Output	hw_vio_1			> 1 gt_txdiffctrl2[19:0]	[H] 0_0000	•	Output	hw_vio_2
> 🗓 gt_txpostcursor[19:0]	[H] 0_0000 👻		Output	hw_vio_1			> 1 gt_txpostcursor2[19:0]	[H] 0_0000	•	Output	hw_vio_2
> 1 gt_txdiffctrl[19:0]	[H] 0_0000 🔻		Output	hw_vio_1			> 1 gt_txprecursor2[19:0]	[H] 0_0000	•	Output	hw_vio_2

4.3. Passive 2m copper cable

2m copper cable을 J20의 QSFP28 #0과 J23의 QSFP28 #1에 삽입한다.

아래 그림처럼 이 경우 둘 중 하나의 루프백 모드를 Far-End PMA loopback 모드로 설정하여야 한다.

v_vios							? 🗆 🖒 >	da	shboard_1				
hw_vio_1							? _ □ ×		hw_vio_2				
Q	≑ + -							tions	Q ¥ ≑ + -				
Name		Value	Ac	ctivity Di	irection	VIO		dop	Name	Value	Activity	Direction	VIO
ι₀ rx_align	ned_led	٠		In	iput	hw_vio_1		boar	Tx_aligned_led2	٠		Input	hw_vio_2
ι, rx_gt_lα	ocked_led	•		In	iput	hw_vio_1		lash	Tx_gt_locked_led2	•		Input	hw_vio_2
ι₀ rx_done	e_led	٠		In	iput	hw_vio_1			"₀ rx_done_led2	۲		Input	hw_vio_2
ີ₀ tx_done	e_led	•		In	put	hw_vio_1			l₀ tx_done_led2	٠		Input	hw_vio_2
ີ₀ rx_busy	/_led	۰		In	iput	hw_vio_1			l₀ rx_busy_led2	٠		Input	hw_vio_2
ີ₀ tx_busy	_led	۰		In	iput	hw_vio_1			l₀ tx_busy_led2	۲		Input	hw_vio_2
ີ₀rx_data	_fail_led	٠		In	put	hw_vio_1			Tx_data_fail_led2	٠		Input	hw_vio_2
ι], lbus_tς	_rx_restart_in	0		0	utput	hw_vio_1			L Ibus_tx_nx_restart_in2	0		Output	hw_vio_2
l₀ send_o	continuous_pkts	0		0	utput	hw_vio_1			↓ send_continuous_pkts2	0		Output	hw_vio_2
ી _e sys_res	set	0		0	utput	hw_vio_1			l₀ sys_reset2	0		Output	hw_vio_2
> 🖫 gt_loop	back_in[11:0]	[H] 924	*	O	utput	hw_vio_1			> 1 gt_loopback_in2[11:0]	[H] 000	*	Output	hw_vio_2
> 14 gt_txpre	ecursor[19:0]	[H] 0_0000	*	0	utput	hw_vio_1			> "l_ gt_txdiffctrl2[19:0]	[H] 0_0000	•	Output	hw_vio_2
> 🖫 gt_txpo	stcursor[19:0]	[H] 0_0000	•	O	utput	hw_vio_1			> 🖫 gt_txpostcursor2[19:0]	[H] 0_0000	•	Output	hw_vio_2
> 1, gt_txdiff	fctrl[19:0]	[H] 0_0000	*	0	utput	hw_vio_1			> 1 gt_txprecursor2[19:0]	[H] 0_0000	*	Output	hw vio 2

QSFP28 #0을 Far-End PMA loopback모드로 설정한 경우



hw_	vios					? 🗆 🖒 X	d	ast	iboard_1		
	hw_vio_1					? _ 🗆 ×		I	hw_vio_2		
tions	Q ≚ ≑ + -						tions		Q ¥ ≑ + -		
dop	Name	Value	Activity	Direction	VIO		a0 p		Name	Value	Activity
boar	Ъ rx_aligned_led	۰		Input	hw_vio_1		boar		Tx_aligned_led2	•	
lash	T_ rx_gt_locked_led	•		Input	hw_vio_1		lash		Tx_gt_locked_led2	•	
	l₀ rx_done_led	۲		Input	hw_vio_1				T_ rx_done_led2	۲	
	l₀ tx_done_led	•		Input	hw_vio_1				↓ tx_done_led2	•	
	l₀ rx_busy_led	۲		Input	hw_vio_1				led2 □_ rx_busy_led2	•	
	l₀ tx_busy_led	۲		Input	hw_vio_1				l₀ tx_busy_led2	•	
	Tx_data_fail_led	۲		Input	hw_vio_1				Tx_data_fail_led2	۲	
	.], lbus_tx_n_restart_in	0		Output	hw_vio_1				lbus_tx_rx_restart_in2	0	
	l₄ send_continuous_pkts	0		Output	hw_vio_1				□ send_continuous_pkts2	0	
	l, sys_reset	0		Output	hw_vio_1				l₀ sys_reset2	0	
	> 1, gt_loopback_in[11:0]	[H] 000	•	Output	hw_vio_1				> 1, gt_loopback_in2[11:0]	[H] 924 🔹	
	> 🖫 gt_txprecursor[19:0]	[H] 0_0000	•	Output	hw_vio_1				> 🖫 gt_txdiffctrl2[19:0]	[H] 0_0000 🔹	
	> 1, gt_txpostcursor[19:0]	[H] 0_0000	*	Output	hw_vio_1				> 1, gt_txpostcursor2[19:0]	[H] 0_0000	
	> 1 gt_txdiffctrl[19:0]	[H] 0_0000	*	Output	hw_vio_1				> 1, gt_txprecursor2[19:0]	[H] 0_0000 🔹	

두 경우 모두 테스트가 실패하고 있다.

IBERT를 이용하여 TX Pattern – PRBS 31-bit, RX Pattern – PRBS 32-bit에 대해서 Errors가 0이 되는 TX Pre-Cursor, TX Post-Cursor, TX Diff Swing 값을 찾아서 cmac_usplus_0_ex_vio_1_2의 VIO에 설정 한다.

TclConsole Messages Serial I/O Links X Serial I/O Scans																		
Q ¥ ≑ +,	a ≭ ∳ +]																	
Name	TX	RX	Status	Bits	Errors	BER	BERT Reset	TX Pattern		RX Pattern		TX Pre-Cursor		TX Post-Cursor		TX Diff Swing		DFE Enabled
Ungrouped Links (0)																		
~ % Link Group 0 (8)							Reset	PRBS 31-bit	~	PRBS 31-bit	~	1.16 dB (00101)	~	6.02 dB (10100)	~	470 mV (00011)	~	✓
% Link 0	MGT_X0Y12/TX	MGT_X0Y12/RX	25.781 Gbps	1.453E11	0E0	6.883E-12	Reset	PRBS 31-bit	~	PRBS 31-bit	~	1.16 dB (00101)	\sim	6.02 dB (10100)	~	470 mV (00011)	\sim	V
% Link 1	MGT_X0Y13/TX	MGT_X0Y13/RX	25.781 Gbps	1.453E11	0E0	6.88E-12	Reset	PRBS 31-bit	~	PRBS 31-bit	~	1.16 dB (00101)	\sim	6.02 dB (10100)	~	470 mV (00011)	~	1
% Link 2	MGT_X0Y14/TX	MGT_X0Y14/RX	25.781 Gbps	1.454E11	0E0	6.88E-12	Reset	PRBS 31-bit	~	PRBS 31-bit	~	1.16 dB (00101)	\sim	6.02 dB (10100)	~	470 mV (00011)	\sim	1
% Link 3	MGT_X0Y15/TX	MGT_X0Y15/RX	25.780 Gbps	1.454E11	0E0	6.879E-12	Reset	PRBS 31-bit	~	PRBS 31-bit	~	1.16 dB (00101)	\sim	6.02 dB (10100)	~	470 mV (00011)	\sim	V
% Link 4	MGT_X0Y16/TX	MGT_X0Y16/RX	25.768 Gbps	1.454E11	0E0	6.878E-12	Reset	PRBS 31-bit	~	PRBS 31-bit	~	1.16 dB (00101)	\sim	6.02 dB (10100)	~	470 mV (00011)	~	1
% Link 5	MGT_X0Y17/TX	MGT_X0Y17/RX	25.781 Gbps	1.454E11	0E0	6.878E-12	Reset	PRBS 31-bit	~	PRBS 31-bit	~	1.16 dB (00101)	\sim	6.02 dB (10100)	~	470 mV (00011)	\sim	V
% Link 6	MGT_X0Y18/TX	MGT_X0Y18/RX	25.781 Gbps	1.454E11	0E0	6.877E-12	Reset	PRBS 31-bit	~	PRBS 31-bit	~	1.16 dB (00101)	~	6.02 dB (10100)	~	470 mV (00011)	~	1
N Link 7	MGT_X0Y19/TX	MGT_X0Y19/RX	25.781 Gbps	1.454E11	0E0	6.877E-12	Reset	PRBS 31-bit	~	PRBS 31-bit	~	1.16 dB (00101)	~	6.02 dB (10100)	~	470 mV (00011)	~	✓

Direction VIO

Input Input

Input

Input

Input Input

Input

Output

Output

Output

Output

Output

Output

Output

hw vio 2

hw_vio_2

hw_vio_2

hw_vio_2 hw_vio_2

hw_vio_2

hw_vio_2

hw_vio_2

hw_vio_2

hw_vio_2

hw_vio_2

hw_vio_2

hw_vio_2

hw_vio_2

hw_	vios					? 🗆 🖒 ×	das	hboard_1				
	hw_vio_1					? _ 🗆 ×		hw_vio_2				
SUO	Q ¥ ♦ + -						ions	Q ¥ ♦ + -				
	Name	Value	Activity	Direction	VIO		d O b	Name	Value	Activity	Direction	VIO
IPOO]₀ rx_aligned_led	•		Input	hw_vio_1		boan	□ rx_aligned_led2	•		Input	hw_vio_2
USP	T_ rx_gt_locked_led	•		Input	hw_vio_1		lash	T_ rx_gt_locked_led2	•		Input	hw_vio_2
	T_ nx_done_led	•		Input	hw_vio_1			□ rx_done_led2	0		Input	hw_vio_2
	l₀ tx_done_led	•		Input	hw_vio_1			l₀ tx_done_led2	•		Input	hw_vio_2
	l₀ nx_busy_led	•		Input	hw_vio_1			l₀ rx_busy_led2	•		Input	hw_vio_2
	□ tx_busy_led	•		Input	hw_vio_1			∃ tx_busy_led2	٠		Input	hw_vio_2
	□ rx_data_fail_led	•		Input	hw_vio_1			□ rx_data_fail_led2	•		Input	hw_vio_2
	lbus_tx_rx_restart_in	0		Output	hw_vio_1			l_ lbus_tx_rx_restart_in2	0		Output	hw_vio_2
	□ send_continuous_pkts	0		Output	hw_vio_1			l₄ send_continuous_pkts2	0		Output	hw_vio_2
	l₀ sys_reset	0		Output	hw_vio_1			l₄ sys_reset2	0		Output	hw_vio_2
	> 🗓 gt_loopback_in[11:0]	[H] 924 ×		Output	hw_vio_1			> 1 gt_loopback_in2[11:0]	[H] 000		Output	hw_vio_2
	> 1 gt_txprecursor[19:0]	[H] 2_94A5 🔹		Output	hw_vio_1			> 1 gt_txprecursor2[19:0]	[H] 2_94A5	•	Output	hw_vio_2
	> 🖫 gt_txpostcursor[19:0]	[H] A_5294 🔹		Output	hw_vio_1			> 🗓 gt_txpostcursor2[19:0]	[H] A_5294	-	Output	hw_vio_2
	> 1 gt_txdiffctrl[19:0]	[H] 1_8C63 👻		Output	hw_vio_1			> 🗓 gt_txdiffctrl2[19:0]	[H] 1_8C63		Output	hw_vio_2

QSFP28 #0을 Far-End PMA loopback모드로 설정한 경우

QSFP28 #1을 Far-End PMA loopback모드로 설정한 경우

hw	_vios					? 🗆 🖒 X	das	hboard_1				
	hw_vio_1					? _ 🗆 ×		hw_vio_2				
ons	Q ≍ ≑ + -						Suo	Q 素 ≑ + -				
d opt	Name	Value	Activity	Direction	VIO		d Opt	Name	Value	Activity	Direction	VIO
ooan	T_ rx_aligned_led	•		Input	hw_vio_1		Doan	T_ rx_aligned_led2	•		Input	hw_vio_2
lash	l₀ rx_gt_locked_led	•		Input	hw_vio_1		lash	locked_led2	•		Input	hw_vio_2
	l₀ rx_done_led	•		Input	hw_vio_1			l₀ rx_done_led2	•		Input	hw_vio_2
	l₀ tx_done_led	•		Input	hw_vio_1			la tx_done_led2	•		Input	hw_vio_2
	l₀ rx_busy_led	•		Input	hw_vio_1			l₀ rx_busy_led2	٠		Input	hw_vio_2
	l₀ tx_busy_led	•		Input	hw_vio_1			led2 led2 led2	۲		Input	hw_vio_2
	🐌 rx_data_fail_led	•		Input	hw_vio_1			l₀ rx_data_fail_led2	٠		Input	hw_vio_2
	lbus_tx_nx_restart_in	0		Output	hw_vio_1			lbus_tx_rx_restart_in2	0		Output	hw_vio_2
	□ send_continuous_pkts	0		Output	hw_vio_1			□ send_continuous_pkts2	0		Output	hw_vio_2
	l₀ sys_reset	0		Output	hw_vio_1			l _e sys_reset2	0		Output	hw_vio_2
	> 🖫 gt_loopback_in[11:0]	[H] 000 🔻		Output	hw_vio_1			> 🐌 gt_loopback_in2[11:0]	[H] 924	•	Output	hw_vio_2
	> 1 gt_txprecursor[19:0]	[H] 2_94A5 🔹		Output	hw_vio_1			> 1 gt_txprecursor2[19:0]	[H] 2_94A5	•	Output	hw_vio_2
	> 1 gt_txpostcursor[19:0]	[H] A_5294 🔹		Output	hw_vio_1			> 🖫 gt_txpostcursor2[19:0]	[H] A_5294	*	Output	hw_vio_2
	> 🗓 gt_txdiffctrl[19:0]	[H] 1_8C63 🔹		Output	hw_vio_1			> 🗓 gt_txdiffctrl2[19:0]	[H] 1_8C63	*	Output	hw_vio_2

IBERT에서 찾은 값으로 테스트가 성공함을 알 수 있다.

4.4. Active optical cable

Active optical cable을 J20의 QSFP28 #0과 J23의 QSFP28 #1에 삽입한다.

Lpmode를	GPIO를	설정하여	해제하여야	하므로	제공된	Vivado	프로젝트
(M_2_2019112	7_NORMAL	_PERST.zip)의	SDK에서 gpio	프로젝트를	실행한다.		

🔍 🔍 со	0M14:115	200baud	- Tera Te	rm VT				
File E	dit Setu	p Contr	ol Wind	low Help				
GPIO	Polle	ed Mod	le Exa	ample 7	lest			
Data	read	from	GPIO	Input	35	is	0x0	
Data	read	from	GPIO	Input	36	is	0x1	
Data	read	from	GPIO	Input	37	is	0x0	
Data	read	from	GPIO	Input	38	is	0x1	
Data	read	from	GPIO	Input	39	is	0x0	
Data	read	from	GPIO	Input	40	is	0x0	
Data	read	from	GPIO	Input	41	is	0x1	
Data	read	from	GPIO	Input	42	is	0x0	
Data	read	from	GPIO	Input	43	is	0x1	
Data	read	from	GPIO	Input	44	is	0x0	
Data	read	from	GPIO	Input	is	0×0		
Succe	essful	lly ra	an GPI	[O Pol]	Led	Mode	Example	Test

다시 한번 cmac_usplus_0_ex_vio_1_2 프로젝트의 bitstream을 다운로드한다.

아래 그림처럼 이 경우 둘 중 하나의 루프백 모드를 Far-End PMA loopback 모드로 설정하여야 한다.

hw	vios					? 🗆 🖒 X	das	shboard_1							
	hw_vio_1							hw_vio_2							
ions	Q ¥ ♦ + -					§ Q ¥ ≑ + =									
d Opt	Name	Value	Activity	Direction	VIO		10 pt	Name	Value	Activity	Direction	VIO			
poar	□ nc_aligned_led	•		Input	hw_vio_1		poar	Tx_aligned_led2	•		Input	hw_vio_2			
lash	l₀ nx_gt_locked_led	•		Input	hw_vio_1		lash	Tx_gt_locked_led2	•		Input	hw_vio_2			
	l₀ nx_done_led	•		Input	hw_vio_1			Tr_done_led2	•		Input	hw_vio_2			
	∃ tx_done_led	•		Input	hw_vio_1			l₀ tx_done_led2	•		Input	hw_vio_2			
	□ nr_busy_led	•		Input	hw_vio_1			l₀ rx_busy_led2	•		Input	hw_vio_2			
	∃ _e tx_busy_led	•		Input	hw_vio_1			l₀ tx_busy_led2	۲		Input	hw_vio_2			
	T_ rx_data_fail_led	•		Input	hw_vio_1			l₀ rx_data_fail_led2	•		Input	hw_vio_2			
	□ lbus_tx_rx_restart_in	0		Output	hw_vio_1			lbus_tx_nx_restart_in2	0		Output	hw_vio_2			
	∃ _e send_continuous_pkts	0		Output	hw_vio_1			l₀ send_continuous_pkts2	0		Output	hw_vio_2			
	l₀ sys_reset	0		Output	hw_vio_1			l₄ sys_reset2	0		Output	hw_vio_2			
	> 1 gt_loopback_in[11:0]	[H] 924 ·		Output	hw_vio_1			> 1 gt_loopback_in2[11:0]	[H] 000 👻		Output	hw_vio_2			
	> 1 gt_txprecursor[19:0]	[H] 0_0000 ·		Output	hw_vio_1			> 1 gt_txprecursor2[19:0]	[H] 0_0000 v		Output	hw_vio_2			
	> "Le gt_txpostcursor[19:0]	[H] 0_0000 ·		Output	hw_vio_1			> 1, gt_txpostcursor2[19:0]	[H] 0_0000 v		Output	hw_vio_2			
	> "L_ gt_txdiffctrl[19:0]	[H] 0_0000 -		Output	hw_vio_1			> 1 gt_txdiffctrl2[19:0]	[H] 0_0000 v		Output	hw_vio_2			

QSFP28 #0을 Far-End PMA loopback모드로 설정한 경우

1w_1	ios					? 🗆 🖾 ×	das	hboard_1							
ſ	hw_vio_1 ? _						hw_vio_2								
	Q ¥ ≑ + -						tions	Q ≚ ≑ + -							
	Name	Value	Activity	Direction	VIO		dOp	Name	Value	Activity	Direction	VIO			
POO	□ rx_aligned_led	•	+	Input	hw_vio_1		boar	□ rx_aligned_led2	•		Input	hw_vio_2			
8	T_ rx_gt_locked_led	•		Input	hw_vio_1		ash] _e nx_gt_locked_led2	•		Input	hw_vio_2			
	l_ rx_done_led	•	+	Input	hw_vio_1			l₀ rx_done_led2	•	+	Input	hw_vio_2			
	la tx_done_led	•		Input	hw_vio_1			la tx_done_led2	0	+	Input	hw_vio_2			
	l₀ rx_busy_led	٠		Input	hw_vio_1			¯₀ π_busy_led2	•		Input	hw_vio_2			
	l₀ tx_busy_led	۲		Input	hw_vio_1			l₀ tx_busy_led2	•		Input	hw_vio_2			
	T_ rx_data_fail_led	۲		Input	hw_vio_1			T_ rx_data_fail_led2	٠		Input	hw_vio_2			
	lbus_tx_nc_restart_in	0		Output	hw_vio_1			↓ Ibus_tx_rx_restart_in2	0		Output	hw_vio_2			
	l _e send_continuous_pkts	0		Output	hw_vio_1			l _e send_continuous_pkts2	0		Output	hw_vio_2			
	l _■ sys_reset	0		Output	hw_vio_1			l, sys_reset2	0		Output	hw_vio_2			
	> 🖫 gt_loopback_in[11:0]	[H] 000	*	Output	hw_vio_1			> 1 gt_loopback_in2[11:0]	[H] 924	-	Output	hw_vio_2			
	> 🖫 gt_txprecursor[19:0]	[H] 0_0000	•	Output	hw_vio_1			> 🗓 gt_txprecursor2[19:0]	[H] 0_0000	•	Output	hw_vio_2			
	> 🖫 gt_txpostcursor[19:0]	[H] 0_0000	•	Output	hw_vio_1			> 🗓 gt_txpostcursor2[19:0]	[H] 0_0000	•	Output	hw_vio_2			
	> 1, gt_txdiffctrl[19:0]	[H] 0_0000	•	Output	hw_vio_1			> 1 gt_txdiffctrl2[19:0]	[H] 0_0000		Output	hw_vio_2			

QSFP28 #1을 Far-End PMA loopback모드로 설정한 경우

5. M.2 테스트

테스트에 이용한 NVMe M.2 SSD는 삼성 SM963 NVMe M.2 SSD 480GB MLC이다.

http://www.11st.co.kr/product/SellerProductDetail.tmall?method=getSellerProductDetail&prdNo=2 406238417&gclid=EAIaIQobChMInYjt16jX5QIVFHZgCh2u5A5qEAYYASABEgLdVvD_BwE&utm_term= &utm_campaign=%B1%B8%B1%DB%BC%EE%C7%CEPC+%C3%DF%B0%A1%C0%DB%BE%F7&utm _source=%B1%B8%B1%DB_PC_S_%BC%EE%C7%CE&utm_medium=%B0%CB%BB%F6

이 NVMe M.2 SSD를 CR-DAISY-M2EXP1(B)-REV1.0 보드 위에 장착한 다음 J25에 연결한다.

제공된 Vivado 프로젝트(<u>M_2_20191127_NORMAL_PERST.zip</u>)의 HDF를 이용하여 <u>Petalinux_DAISY_설정_20191126.docx</u>를 참조하여 SD 부트나 QSPI 부트 리눅스 이미지를 생성한 다.

부팅시 LED0가 점등하는지 확인한다. PCle 링크가 정상적으로 설정되면 LED가 켜진다.

리눅스에 로그인하여 lspci로 링크를 확인한다.

root@daisy:~# lspci
0000:00:00.0 PCI bridge: Xilinx Corporation Device 9134
0000:01:00.0 Non-Volatile memory controller: Samsung Electronics Co Ltd NVMe SSD
Controller SM961/PM961
0001:00:00.0 PCI bridge: Xilinx Corporation Device 9134
root@daisy:~#

lsblk로 SSD가 블락 디바이스로 인식되는지 확인한다.

root@daisy:	~# lsblk					
NAME	MAJ:MIN	RM	SIZE	RO	TYPE	MOUNTPOINT
mtdblock0	31:0	0	36M	0	disk	
mtdblock1	31:1	0	1M	0	disk	
mtdblock2	31:2	0	68M	0	disk	
mtdblock3	31:3	0	11M	0	disk	
mmcblk0	179:0	0	14.9G	0	disk	
-mmcblk0p1	179:1	0	1G	0	part	/run/media/mmcblk0p1
`-mmcblk0p2	179:2	0	6.5G	0	part	/run/media/mmcblk0p2
nvme0n1	259:0	0	447.1G	0	disk	
root@daisy:	~#					

```
fdisk /dev/nvme0n1으로 파티션을 생성한다.
root@daisy:~# fdisk /dev/nvme0n1
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.
Device does not contain a recognized partition table.
Created a new DOS disklabel with disk identifier 0xcb728903.
Command (m for help): n
Partition type
  p primary (0 primary, 0 extended, 4 free)
  e extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1):
First sector (2048-937703087, default 2048):
Last sector, +sectors or +size{K,M,G,T,P} (2048-937703087, default 937703087):
Created a new partition 1 of type 'Linux' and of size 447.1 GiB.
Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
[ 1883.709806] nvme0n1: p1
Syncing disks.
```

```
root@daisy:~#
```

lsblk로 새로 생성한 파티션 이름을 확인한다.

root@daisy:~# lsblk										
NAME	MAJ:MIN	RM	SIZE	RO	TYPE	MOUNTPOINT				
mtdblock0	31:0	0	36M	0	disk					
mtdblock1	31:1	0	1M	0	disk					
mtdblock2	31:2	0	68M	0	disk					
mtdblock3	31:3	0	11M	0	disk					
mmcblk0	179:0	0	14.9G	0	disk					
-mmcblk0p1	179:1	0	1G	0	part	/run/media/mmcblk0p1				
`-mmcblk0p2	179:2	0	6.5G	0	part	/run/media/mmcblk0p2				
nvme0n1	259:0	0	447.1G	0	disk					
`-nvme0n1p1	259:1	0	447.1G	0	part					
root@daisy:^	~#									

새로운 파티션에 파일 시스템을 생성한다. root@daisy:~# mkfs -t ext2 /dev/nvme0n1p1 mke2fs 1.44.3 (10-July-2018) Discarding device blocks: done Creating filesystem with 117212630 4k blocks and 29310976 inodes Filesystem UUID: 15a30903-8a05-4beb-9971-21cf586f1caf Superblock backups stored on blocks: 32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208, 4096000, 7962624, 11239424, 20480000, 23887872, 71663616, 78675968, 102400000 Allocating group tables: done Writing inode tables: done Writing superblocks and filesystem accounting information: done root@daisy:~#

디렉토리를 만들고 SSD를 mount한다.

root@daisy:~# mkdir /media/nvme root@daisy:~# mount /dev/nvme0n1p1 /media/nvme root@daisy:~# cd /media/nvme root@daisy:/media/nvme# vi test.txt

파일을 생성하고 리부트하여 보존되는지 확인한다.



root@daisy:/media/nvme# ls -al total 24 drwxr-xr-x 3 root root 4096 Nov 28 09:23 . drwxr-xr-x 3 root root 60 Nov 28 09:20 .. drwx----- 2 root root 16384 Nov 28 09:18 lost+found -rw-r--r-- 1 root root 21 Nov 28 09:23 test.txt root@daisy:/media/nvme#