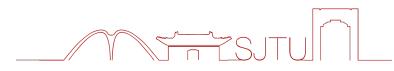
CALICE Collaboration Meeting

CALI(CO



Progress of Fast Timing Electronics with PETIROC2B for SDHCAL

Qiu-Ping Shen, Weihao Wu, Haijun Yang Stephane Callier and Imad Laktineh





3/24-3/26 2021

饮水思源•爱国荣校



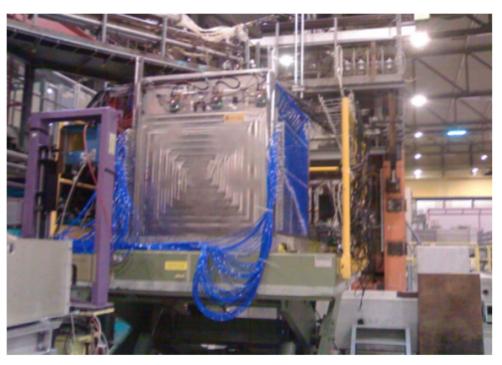
Outline

- Introduction of SDHCAL prototype
- Motivation of using timing information
- Prototype of Front End Electronics (FEE)
- Design of data acquisition system
- Brief summary and future plan



Introduction of SDHCAL prototype

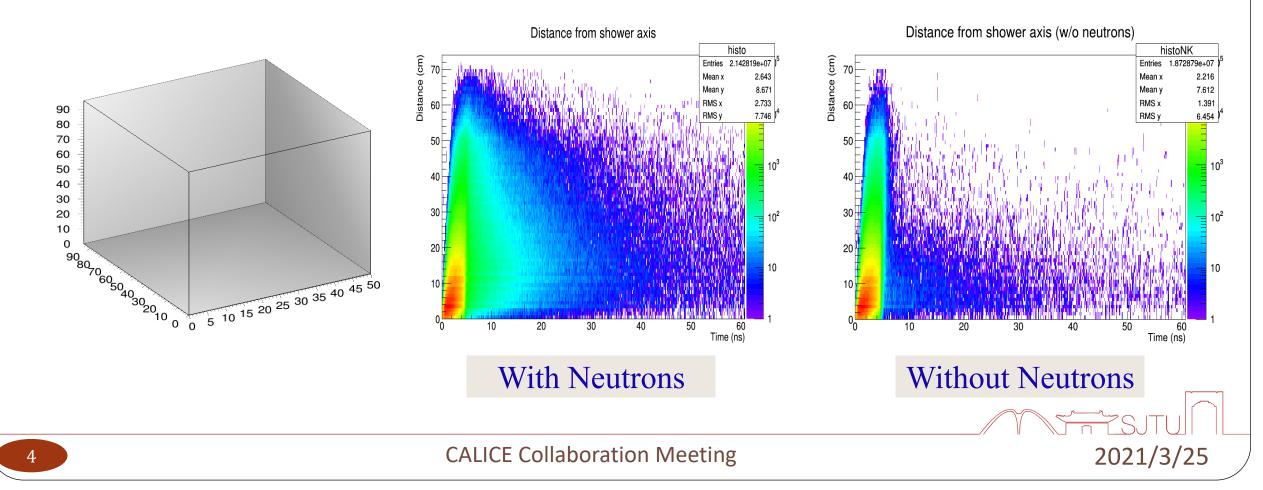
- Semi-Digital Hadronic CALorimeter technological prototype (SDHCAL)
- Igh granularity calorimeter based on Glass RPC (cell size 1cm × 1cm)
- Hits associated to three thresholds:
 - 1st threshold = 110fC
 - 2nd threshold = 5pC
 - 3rd threshold = 15pC
 - \rightarrow Semi-digital readout (0, 1, 2)





Motivation using timing information

Timing could be an important factor to identify delayed neutron and better reconstruct their energy.



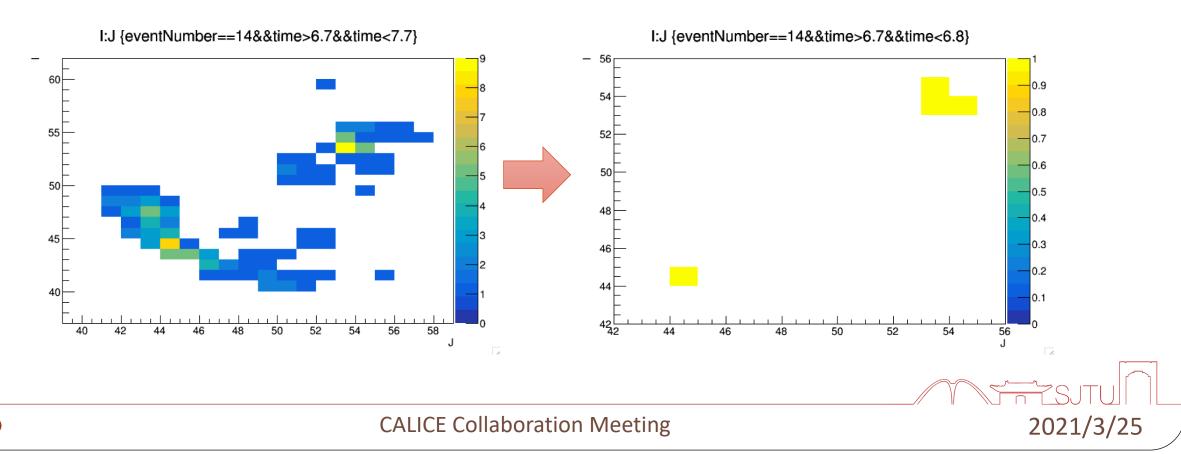


Timing information

Time information can be very helpful to separate close by showers and reduce the confusion for a better PFA application.

1 ns resolution

100ps resolution





Introduction of PETIROC

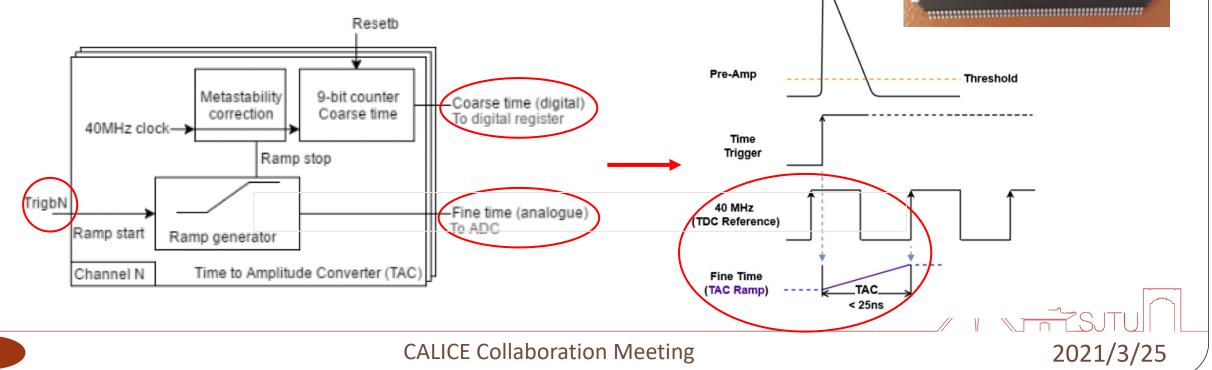
- Time measurement
 - Coarse time is from a counter
 - Fine time by interpolating 40MHz
 - Jitter ~18 ps RMS on trigger output Variable time shaper

(4 photoelectrons injected)



- 32chs input connected
 - with PAD (readout unit)

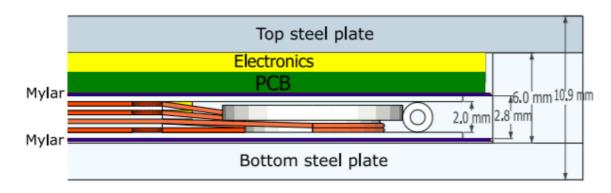






How to achieve fast timing measurement

- Purpose: => Identify neutral and charged hadrons
 Position, Energy and Timing => 5D HCAL
- Adding MRPC layers in the SDHCAL
- Front-end board for MRPC readout
 - Charge and timing measurement
 - High resolution timing measurement





✓ First step:

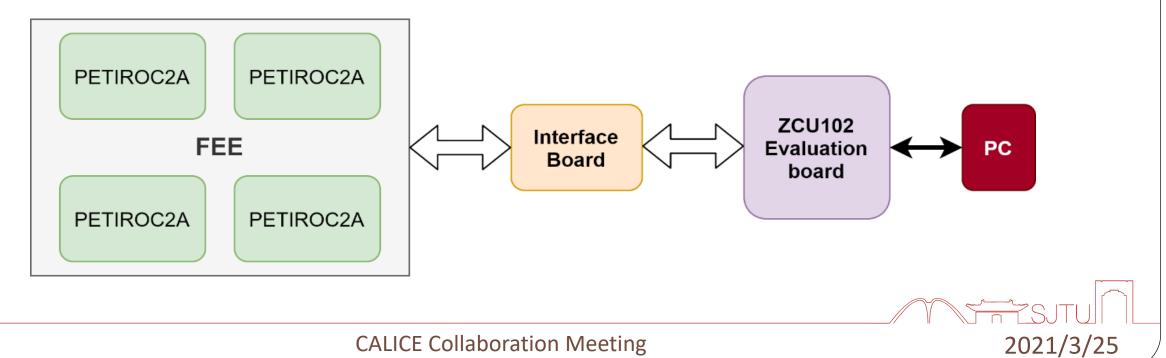
Design a front-end prototype board with four PETIROC2B chips

Second step:
Build the 1m×1m PETIROC2B FEE



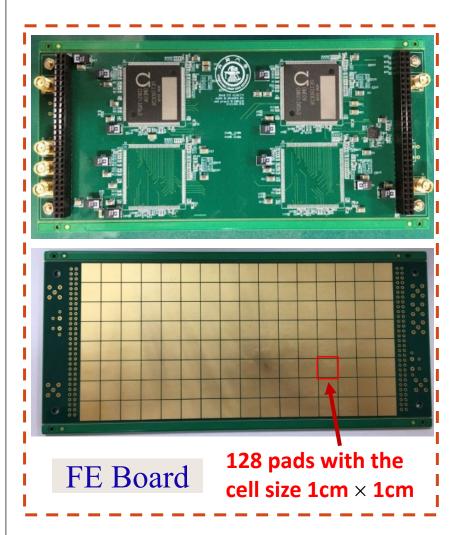
Prototype of timing electronics

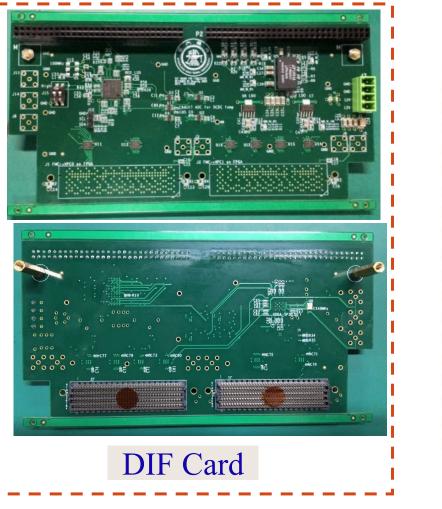
- The FEE prototype includes four PETIROC chips, 128 readout pads on the PCB bottom side.
- Detector Interface(DIF) card was designed to connect FEB and FPGA board
 - Data transmission, power rail and clock source.
- The DAQ system should be developed to transfer data between FPGA and PC.





Hardware of prototype





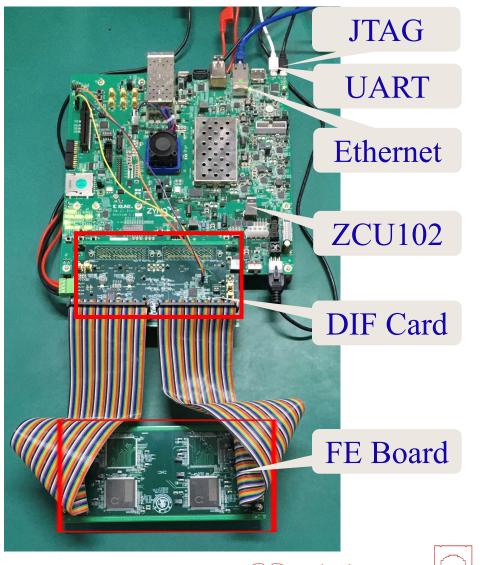


CALICE Collaboration Meeting



Test system and setup

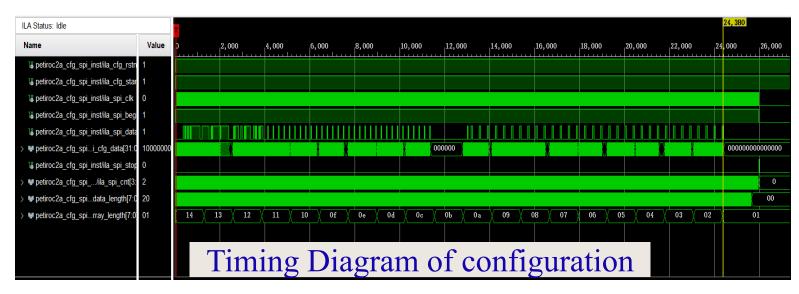
- The test platform has been setup.
- FEB, DIF card, DAQ based on ZCU102
- Status of the platform:
 - Configuration of PETIROC
 - Data transmission between FEB, FPGA and PC
 - Performance test of PETIROC chips, timing measurement

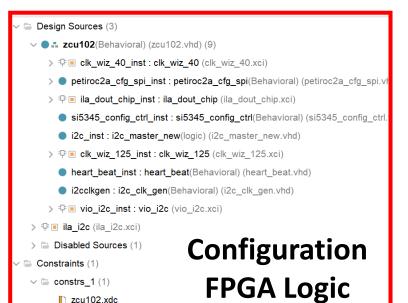




PETIROC chips configuration

- 648 bits data with SPI method is sent to Shift Register inside PETIROC.
- PETIROC configuration works well.
 - All bias voltage values are correct.
 - Output data has been checked, after sending trigger signals



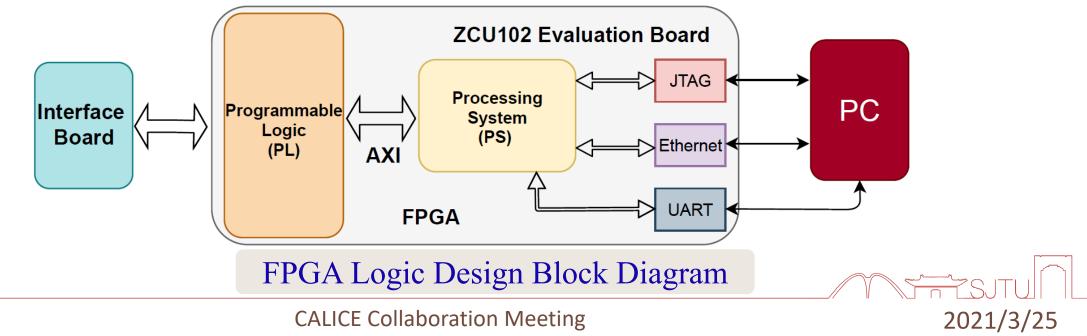


Bias Voltage	Value(V)
vref_inpdac	0.989
vref_time	1.664
vref_charge	0.976
vref_tdc	0.133
vref_adc	0.961
vref_time_pad	1.658



Data acquisition system design

- System is based on Xilinx ZCU102(FPGA) that contains Processing System(PS) and Programmable Logic(PL).
- Embedded design (SDK) in ZCU102(PS) side
 - The UART communication of FPGA and PC
 - Ethernet communication between FPGA and PC used to transfer data
 - Data transmission with AXI Bus between PS and PL, inside of ZCU102





- The embedded design in ZCU102(PS side) contains:
 - Serial port communication (UART)
 - Ethernet communication (TCP/IP)
- WART test in PS side:
 - Hardware only needs
 Processing System part on ZCU102.
 - Run the C++ program on the hardware platform.
 - Information is printed on the tool window through UART port.

<pre>#include <stdio.h> #include <stdio.h> #include <stdio.h> #include "lwip/err.h" #include "lwip/err.h" #include "lwip/err.h" #include "lwip/err.h" #include "lwip/err.h" #include "sil_printf.h" #include state tcp_contected_pob = NULL; #include state tcp_contected_pob = Silent"; #include state tcp_contected_pob *tpcb = connected_pob; #if (ITx_SIZE < tcp_sndbuf(tpcb)) { //#Imtz data for sending (but does not send it immediately). #include state sta</stdio.h></stdio.h></stdio.h></pre>		
<pre># #include "lwip/err.h" #include "lwip/err.h" #include "lwip/err.h" #include "lwip/err.h" #include "sil_cache.h" #include "sil_cache.h" #include "sil_cache.h" #include "sil_cache.h" #include "silep.h" #define TX_SIZE 102 static struct tcp_cb*connected_pcb = NULL; unsigned client_connected = 0; ///tatic Global Function, blind for external file u_char data[TX_SIZE] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}; u_char data[TX_SIZE] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}; u_char data[TX_SIZE] = Hello World! Successfully Send Word From PS Client"; if int send_data() { err_t err; struct tcp_pcb *tpcb = connected_pcb; if (Itpcb) return -1; //jHmSzk#zMarkEgEATA-TZzk@zhr区maksongHkg if (rx_ISIZE < tcp_sndbuf(tpcb)) { //white data for sending (but does not send it immediately). err = tcp_write(tpcb, data, TX_SIZE, 1); if (rx_ISIZE = NULL; return -1; } //Find out what we can send and send it err = tcp_output(tpcb); </pre>	1	<pre>#include <stdio.h></stdio.h></pre>
<pre>i #include "lwip/err.h" #include "lwip/err.h" #include "lwip/err.h" #include "lwip/st.h" #include "xil_printf.h" #include "sil_printf.h" #include #includ</pre>	2	<pre>#include <string.h></string.h></pre>
<pre>5 #include "lwig/tcp.h" #include "lwig/tcp.h" #include "xil_cache.h" #include "xil_printf.h" #include "xil_printf.h" #include "sleep.h" # define TX_SIZE 102 5 static struct tcp.pcb*connected_pcb = NULL; unsigned client_connected = 0; //static Global Function, blind for external file unit tcp_trans_done = 0; //u_char data[TX_SIZE] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}; u_char data[TX_SIZE] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}; u_char data[TX_SIZE] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}; u_char data[TX_SIZE] = thello World! Successfully Send Word From PS Client"; i int send_data() { err_t err; struct tcp_pcb *tpcb = connected_pcb; if (!tpcb) return -1; //jHm%z\ksymExt@g&Ea^+Tg\ksymExmapscond it immediately). err = tcp_write(tpcb, data, TX_SIZE, 1); if (err != <i>ERR_OX</i>) { xil_printf("txperf: Error on tcp_write: %d\r\n", err); connected_pcb = NULL; return -1; } //Find out what we can send and send it err = tcp_output(tpcb);</pre>		
<pre>5 #include "lwipopts.h" #include "xil_cache.h" #include top.h" #include top.h"</pre>		
<pre>7 #include "xil_cache.h" #include "xil_printf.h" #include "xil_printf.h" #include "xil_printf.h" #define TX_SIZE 102 static struct tcp_pcb*connected_pcb = NULL; unsigned client_connected = 0; //Static Global Function, blind for external file uint tcp_trans_done = 0; //U_char data[TX_SIZE] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}; U_char data[TX_SIZE] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}; U_char data[TX_SIZE] = "Hello World! Successfully Send Word From PS Client"; //J in send_data() { err_t err; struct tcp_pcb *tpcb = connected_pcb; if (!tpcb) return -1; //JMIKJ2&\$MEK tcp_BarArFX; Big@rtCmlosing (but does not send it immediately). err = tcp_write(tpcb, data, TX_SIZE, 1); if (err != ERR_OX) { xil_printf("txperf: Error on tcp_write: %d\r\n", err); connected_pcb = NULL; return -1; //Find out what we can send and send it err = tcp_output(tpcb); </pre>		
<pre>#include "sleep.h" #define TX_SIZE 102 static struct tcp_pcb*connected_pcb = NULL; unsigned client_connected = 0; //Static Global Function, blind for external file uint tcp_trans_done = 0; //u_char data[TX_SIZE] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}; u_char data[TX_SIZE] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}; u_char data[TX_SIZE] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}; u_char data[TX_SIZE] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}; u_char data[TX_SIZE] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}; u_char data[TX_SIZE] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}; u_char data[TX_SIZE] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}; u_char data[TX_SIZE] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}; u_char data[TX_SIZE] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}; //Jm setum -1; //Jm setum -1; //Jm gdds data() //Find out what we can send and send it err = tcp_output(tpcb); //Find out what we can send and send it err = tcp_output(tpcb);</pre>	2	#include "Iwipopts.h"
<pre>#include "sleep.h" #define TX_SIZE 102 static struct tcp_pcb*connected_pcb = NULL; unsigned client_connected = 0; //Static Global Function, blind for external file uint tcp_trans_done = 0; //u_char data[TX_SIZE] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}; u_char data[TX_SIZE] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}; u_char data[TX_SIZE] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}; u_char data[TX_SIZE] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}; u_char data[TX_SIZE] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}; u_char data[TX_SIZE] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}; u_char data[TX_SIZE] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}; u_char data[TX_SIZE] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}; u_char data[TX_SIZE] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}; //Jm setum -1; //Jm setum -1; //Jm gdds data() //Find out what we can send and send it err = tcp_output(tpcb); //Find out what we can send and send it err = tcp_output(tpcb);</pre>	<u>_</u>	#include "xil_cache.h" PS CODE
<pre>#define TX_SIZE 102 static struct tcp_pcb*connected_pcb = NULL; unsigned Client_connected = 0; //Static Global Function, blind for external file uint tcp_trans_done = 0; //u_char data[TX_SIZE] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}; u_char data[TX_SIZE] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}; u_char data[TX_SIZE] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}; u_char data[TX_SIZE] = Hello World! Successfully Send Word From PS Client"; ern_t ern; struct tcp_pcb *tpcb = connected_pcb; if (ltpcb) return -1; //#IMS2&\$MatkEgEASATS2&\$Mat</pre>		
<pre>#define TX_SIZE 102 #define TX_SIZE 102 #define TX_SIZE 102 #define TX_SIZE 102 #define TX_SIZE clobal Function, blind for external file #diverse for the standard of the</pre>		#include Sicepin
<pre>static struct tcp_pcb*connected_pcb = NULL; unsigned client_connected = 0; //Static Global Function, blind for external file uint tcp_trans_done = 0; //u_char data[TX_SIZE] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}; u_char data[TX_SIZE] = {0, 1, 2, 3, 4, 5, 7, 7, 8, 9}; u_char data[TX_SIZE] = {0, 1, 2, 3, 4, 5, 7, 7, 8, 9}; u_char data[TX_SIZE] = {0, 1, 2, 3, 4, 5, 7, 7, 8, 9}; u_char data[TX_SIZE] = {0, 1, 2, 3, 4, 5, 7, 7, 8, 9}; u_char data[TX_SIZE] = {0, 1, 2, 3, 4, 5, 7, 7, 8}; u_char data[TX_SIZE] = {0, 1, 3, 5, 7, 7, 7, 8}; u_char data[TX_SIZE] = {0, 1, 3, 5, 7, 7, 7, 8}; u_char data[TX_SIZE] = {0, 1, 3,</pre>		#define TX SI7E 102
<pre>1 unsigned client_connected = 0; //Static Global Function, blind for external file uint tcp_trans_done = 0; //u_char data[TX_SIZE] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}; u_char data[TX_SIZE] = "Hello World! Successfully Send Word From PS Client"; u_char data[TX_SIZE] = "Hello World! Successfully Send Word From PS Client"; int send_data() err_t err; struct tcp_pcb *tpcb = connected_pcb; if (!tpcb) return -1; //Alm 发送数据长度是否小于发送缓冲区剩余可用长度 if (!TX_SIZE < tcp_sndbuf(tpcb)) { //Write data for sending (but does not send it immediately). err = tcp_write(tpcb, data, TX_SIZE, 1); if (err != <i>ERR_OX</i>) { xil_printf("txperf: Error on tcp_write: %d\r\n", err); connected_pcb = NULL; return -1; //Find out what we can send and send it err = tcp_output(tpcb); //Find out what we can send and send it err = tcp_output(tpcb); } </pre>	>	
<pre>i unsigned client_connected = 0; //static Global Function, blind for external file uint tcp_trans_done = 0; //u_char data[TX_SIZE] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}; u_char data[TX_SIZE] = "Hello World! Successfully Send Word From PS Client"; u_char data[TX_SIZE] = "Hello World! Successfully Send Word From PS Client"; u_char data[TX_SIZE] = "Hello World! Successfully Send Word From PS Client"; u_char data[TX_SIZE] = "Hello World! Successfully Send Word From PS Client"; u_char data[TX_SIZE] = "Hello World! Successfully Send Word From PS Client"; if ist send_data() { for trans_data() if (!tpcb) return -1; //J#Jm发送数据长度是否小于发送缓冲区剩余可用长度 if (TX_SIZE < tcp_sndbuf(tpcb)) { //Write data for sending (but does not send it immediately). err = tcp_write(tpcb, data, TX_SIZE, 1); if (err != <i>ERR_OK</i>) { xil_printf("txperf: Error on tcp_write: %d\r\n", err); connected_pcb = NULL; return -1; } //Find out what we can send and send it err = tcp_output(tpcb); } } } //Find out what we can send and send it err = tcp_output(tpcb); } } } //End out what we can send and send it err = tcp_output(tpcb); } } } //End out what we can send and send it err = tcp_output(tpcb); } } } //End out what we can send and send it } } //End out what we can send and send it err = tcp_output(tpcb); } } } //End out what we can send and send it } } //End output (tpcb); } } //End output (tpcb); } //End output (tpcb); } //End output (tpcb); } //End output (tpcb); } //End output (tpcb); } //End output (tpcb); } //End output (tpcb); //End output (tpcb); } //End output (tpcb); //End output (tpcb); //End output (tpcb); //End output (tpcb); //End output (tpcb); //En</pre>	3	<pre>static struct tcp_pcb*connected_pcb = NULL;</pre>
<pre>ivint tcp_trans_done = 0; //u_char data[TX_SIZE] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}; u_char data[TX_SIZE] = "Hello World! Successfully Send Word From PS Client"; int send_data() err_terr; struct tcp_pcb *tpcb = connected_pcb; if (ltpcb) return -1; //判断发送数据长度是否小于发送缓冲区剩余可用长度 if (TX_SIZE < tcp_sndbuf(tpcb)) { //Write data for sending (but does not send it immediately). err = tcp_write(tpcb, data, TX_SIZE, 1); if (err != ERR_OK) { xil_printf("txperf: Error on tcp_write: %d\r\n", err); connected_pcb = NULL; return -1; } //Find out what we can send and send it err = tcp_wutput(tpcb);</pre>	1	
<pre>//har data[TX_SIZE] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}; u_char data[TX_SIZE] = "Hello World! Successfully Send Word From PS Client"; i struct tcp_pcb *tpcb = connected_pcb; if (ltpcb) return -1; //判断发送数据长度是否小于发送缓冲区剩余可用长度 if (TX_SIZE < tcp_sndbuf(tpcb)) { //Write data for sending (but does not send it immediately). err = tcp_write(tpcb, data, TX_SIZE, 1); if (err != ER_OX) { xil_printf("txperf: Error on tcp_write: %d\r\n", err); connected_pcb = NULL; return -1; //Find out what we can send and send it err = tcp_output(tpcb);</pre>		//Static Global Function, blind for external file
<pre>u_char data[TX_SIZE] = "Hello World! Successfully Send Word From PS Client"; int send_data() err_t err; struct tcp_pcb *tpcb = connected_pcb; if (!tpcb) return -1; //判断发送数据长度是否小于发送缓冲区剩余可用长度 if (TX_SIZE < tcp_sndbuf(tpcb)) { //Write data for sending (but does not send it immediately). err = tcp_write(tpcb, data, TX_SIZE, 1); if (err != <i>ERR_OX</i>) { xil_printf("txperf: Error on tcp_write: %d\r\n", err); connected_pcb = NULL; return -1; } //Find out what we can send and send it err = tcp_output(tpcb);</pre>	5	uint tcp_trans_done = 0;
<pre>u_char data[TX_SIZE] = "Hello World! Successfully Send Word From PS Client"; int send_data() err_t err; struct tcp_pcb *tpcb = connected_pcb; if (!tpcb) return -1; //判断发送数据长度是否小于发送缓冲区剩余可用长度 if (TX_SIZE < tcp_sndbuf(tpcb)) { //Write data for sending (but does not send it immediately). err = tcp_write(tpcb, data, TX_SIZE, 1); if (err != <i>ERR_OX</i>) { xil_printf("txperf: Error on tcp_write: %d\r\n", err); connected_pcb = NULL; return -1; } //Find out what we can send and send it err = tcp_output(tpcb);</pre>	7	
<pre>int send_data() { int send_data() { int send_data() for an analysis of the second se</pre>		
<pre>int send_data() { err_t err; struct tcp_pcb *tpcb = connected_pcb; if (ltpcb) return -1; //判断发送数据长度是否小于发送缓冲区剩余可用长度 if (Tx_SIZE < tcp_sndbuf(tpcb)) { //Write data for sending (but does not send it immediately). err = tcp_write(tpcb, data, TX_SIZE, 1); if (err != <i>ERR_OK</i>) { xil_printf("txperf: Error on tcp_write: %d\r\n", err); connected_pcb = NULL; return -1; //Find out what we can send and send it err = tcp_output(tpcb); //Find out what we can send and send it err = tcp_output(tpcb); //Find out what we can send and send it err = tcp_output(tpcb); //Find out what we can send and send it err = tcp_output(tpcb); //Find out what we can send and send it err = tcp_output(tpcb); //Find output(tpcb); // Find output(tpcb); // Find output(tpcb); // Find output(tpc</pre>		u_char data[IX_SIZE] = Hello world! Successfully Send word From PS Client;
<pre>2 { err_t err; struct tcp_pcb *tpcb = connected_pcb; if (ltpcb) return -1; //#IMS发送数据长度是否小于发达缓冲区剩余可用长度 if (TX_SIZE < tcp_sndbuf(tpcb)) { //Write data for sending (but does not send it immediately). err = tcp_write(tpcb, data, TX_SIZE, 1); if (err != ER_OX) { xil_printf("txperf: Error on tcp_write: %d\r\n", err); connected_pcb = NULL; return -1; } //Find out what we can send and send it err = tcp_output(tpcb); </pre>		int send data()
<pre>erc_t err; struct tcp.pcb *tpcb = connected_pcb; if (ltpcb) return -1; //判断发送数据长度是否小于发送缓冲区剩余可用长度 if (TX_SIZE < tcp_sndbuf(tpcb)) { //write data for sending (but does not send it immediately). err = tcp_write(tpcb, data, TX_SIZE, 1); if (err != ERR_OK) { xil_printf("txperf: Error on tcp_write: %d\r\n", err); connected_pcb = NULL; return -1; } //Find out what we can send and send it err = tcp_output(tpcb);</pre>		
<pre>struct tcp_pcb *tpcb = connected_pcb; if (!tpcb) return -1; //判断发送数据长度是否小于发送缓冲区剩余可用长度 if (TX_SIZE < tcp_sndbuf(tpcb)) { //Write data for sending (but does not send it immediately). err = tcp_write(tpcb, data, TX_SIZE, 1); if (err != <i>ERR_OX</i>) { xil_printf("txperf: Error on tcp_write: %d\r\n", err); connected_pcb = NULL; return -1; } //Find out what we can send and send it err = tcp_output(tpcb);</pre>		
<pre>if (!tpcb) return -1; //判断发送数据长度是否小于发送缓冲区剩余可用长度 if (Tx_SIZE < tcp_sndbuf(tpcb)) { //Write data for sending (but does not send it immediately). err = tcp_write(tpcb, data, TX_SIZE, 1); if (err != <i>ERR_OK</i>) { xil_printf("txperf: Error on tcp_write: %d\r\n", err); connected_pcb = NULL; return -1; //Find out what we can send and send it err = tcp_output(tpcb); //Find out what we can send and send it err = tcp_output(tpcb); //Find out what we can send and send it err = tcp_output(tpcb); //Find out what we can send and send it err = tcp_output(tpcb); //Find output(tpcb); // Find output(tpcb);</pre>		
<pre>return -1; //判断发送数据长度是否小于发送缓冲区剩余可用长度 if (TX_SIZE < tcp_sndbuf(tpcb)) { //Wite data for sending (but does not send it immediately). err = tcp_write(tpcb, data, TX_SIZE, 1); if (err != ERR_OK) { xil_printf("txperf: Error on tcp_write: %d\r\n", err); connected_pcb = NULL; return -1; } //Find out what we can send and send it err = tcp_output(tpcb); </pre>	5	
<pre>//判断发送数据长度是否小于发送缓冲区剩余可用长度 if (TX_SIZE < tcp_sndbuf(tpcb)) { //Write data for sending (but does not send it immediately). err = tcp_write(tpcb, data, TX_SIZE, 1); if (err != ERR_OK) { xil_printf("txperf: Error on tcp_write: %d\r\n", err); connected_pcb = NULL; return -1; //Find out what we can send and send it err = tcp_output(tpcb); } </pre>		if (!tpcb)
<pre>//判断发送数据长度是否小于发送缓冲区剩余可用长度 if (TX_SIZE < tcp_sndbuf(tpcb)) { //Write data for sending (but does not send it immediately). err = tcp_write(tpcb, data, TX_SIZE, 1); if (err != <i>ER</i>_OX) { xil_printf("txperf: Error on tcp_write: %d\r\n", err); connected_pcb = NULL; return -1; } //Find out what we can send and send it err = tcp_output(tpcb);</pre>		return -1;
<pre>if (TX_SIZE < tcp_sndbuf(tpcb)) { //Write data for sending (but does not send it immediately). err = tcp_write(tpcb, data, TX_SIZE, 1); if (err != ERR_OK) { xil_printf("txperf: Error on tcp_write: %d\r\n", err); connected_pcb = NULL; return -1; } //Find out what we can send and send it err = tcp_output(tpcb); </pre>		()到底会谈教授长帝国家中主会法院冲区到全国任帝
<pre>1 //Write data for sending (but does not send it immediately). err = tcp_write(tpcb, data, TX_SIZE, 1); 3 if (err != ERR_OX) { 4 xil_printf("txperf: Error on tcp_write: %d\r\n", err); 5 connected_pcb = NULL; 6 return -1; 7 } 9 //Find out what we can send and send it 9 err = tcp_output(tpcb);</pre>		
<pre>err = tcp_write(tpcb, data, TX_SIZE, 1); if (err = ERR_OK) { xil_printf("txperf: Error on tcp_write: %d\r\n", err); connected_pcb = NULL; return -1; } //Find out what we can send and send it err = tcp_output(tpcb);</pre>		
<pre>if (err]= ERR_OK) { xil_printf("txperf: Error on tcp_write: %d\r\n", err); connected_pcb = NULL; return -1; } //Find out what we can send and send it err = tcp_output(tpcb);</pre>		
<pre>connected_pcb = NULL; return -1; } //Find out what we can send and send it err = tcp_output(tpcb);</pre>		
<pre>5 return -1; 7 } 9 //Find out what we can send and send it 9 err = tcp_output(tpcb);</pre>	ı.	<pre>xil_printf("txperf: Error on tcp_write: %d\r\n", err);</pre>
<pre>} //Find out what we can send and send it err = tcp_output(tpcb);</pre>		connected_pcb = NULL;
<pre>//Find out what we can send and send it err = tcp_output(tpcb);</pre>		
<pre>//Find out what we can send and send it err = tcp_output(tpcb);</pre>		}
err = tcp_output(tpcb);		(Piel ach data an and and and it
<pre>xil printf("typerf: Error on tcp output: %d\r\n",err);</pre>		
return -1;		
1 }		
1	:	1

端 口 COM4

数据位 8

停止位 1 流 控 None

接收设置

• ASCII

☑ 自动换行□ 显示发送

▼ 显示时间

□ 自动重发 1000

·发送设置 • ASCII ○ Hex

Hex

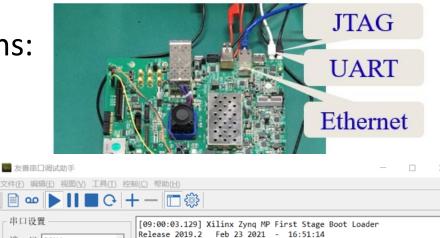
÷ms

COM4 OPENED, 115200, 8, NONE, 1, OFF Rx: 449 Bytes

sad

校验位 None

波特率 115200



On Host: Run \$iperf -s -i 5 -w 2M

UART Output

[09:00:09.261] autonegotiation complete link speed for phy address 12: 1000

Start PHY autonegotiation

PMU-FW is not running, certain applications may not be supported. [09:00:05.247] TCP client connecting to 192.168.1.100 on port 5001

UART Debug Assistant

Tx: 0 Byte

[09:00:05.253] Waiting for PHY to complete autonegotiation.

Colla	horation	Meeting	
COLL	OO(A)O(1)	NEELINS	

UART communication test

2021/3/25

发送

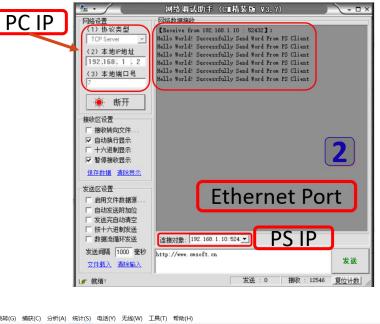
DAQ system -- Ethernet

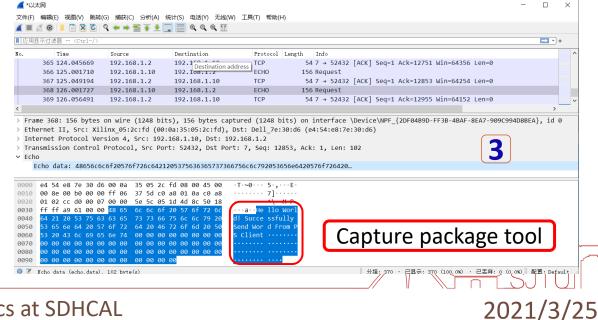
- Ethernet communication test:
 - **1** PS program sent the data to PC.

//u_char data[TX_SIZE] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}; u char data[TX SIZE] = "Hello World! Successfully Send Word From PS Client";

- UART used to print PS information (IP, Sub netmask, Gateway)
- 2 Ethernet port connected with PC, used to test the transmission and print the transfer data.
- 3 Capture package tool can grab the transmit information.

Section 20102(PS) and PC can successfully communicate over ethernet.





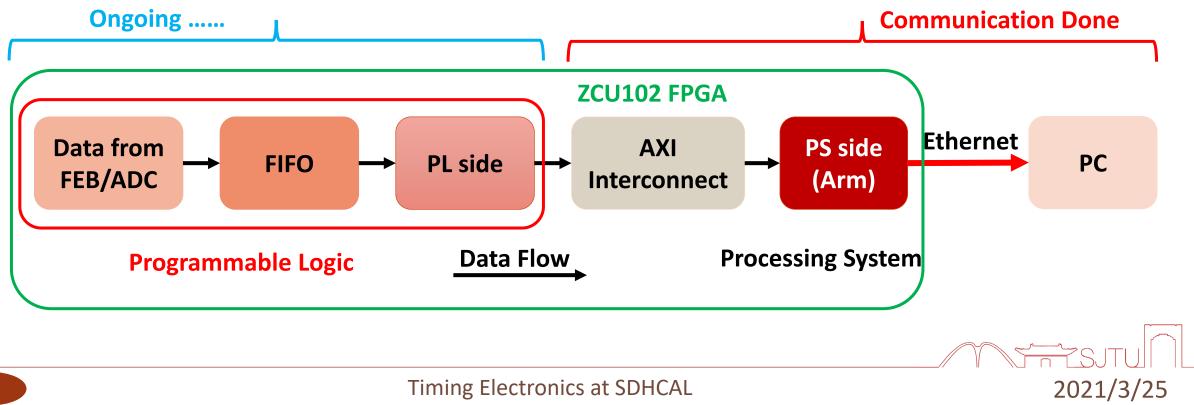


🚄 *以太网



Design of DAQ system

- UART communication works well.
- Ethernet transmission between ZCU102(PS) and PC is completed.
- The communication between FE Board, FIFO and ZCU102(PL) is ongoing.
- The configuration of PETIROC chips over ethernet will be included.



Brief summary and future plan

Summary:

- The timing electronics have been designed and manufactured.
- Test platform and setup for PETIROC have been constructed.
- PETIROC chips can be successfully configured.
- Sthernet transmission between FPGA and PC has been realized.
- The design of DAQ system is still under the development.

Future Plan:

- Sontinue the design of DAQ system based FPGA ZCU102 to achieve the data transmission and slow control over the ethernet communication.
- Perform the signal injection test for PETIROC FE board (Timing and Charge)

2021/3/25



Thanks for your attention!



CALICE Collaboration Meeting



Backup Slides



CALICE Collaboration Meeting

18



Introduction of PETIROC chip

Time measurement with 10bits TDC interpolating 40MHz coarse time

- Scharge measurement (Q>50fC) with 10bits DAC
- Voltage input amplifier, 2000hm matching
- High bandwidth preamp (GBWP> 1.2 GHz)
- PETIROC parameters:
 - One chip with 32-channels and mixed analog/digital
 - The 32chs input connected with PAD (detector unit)

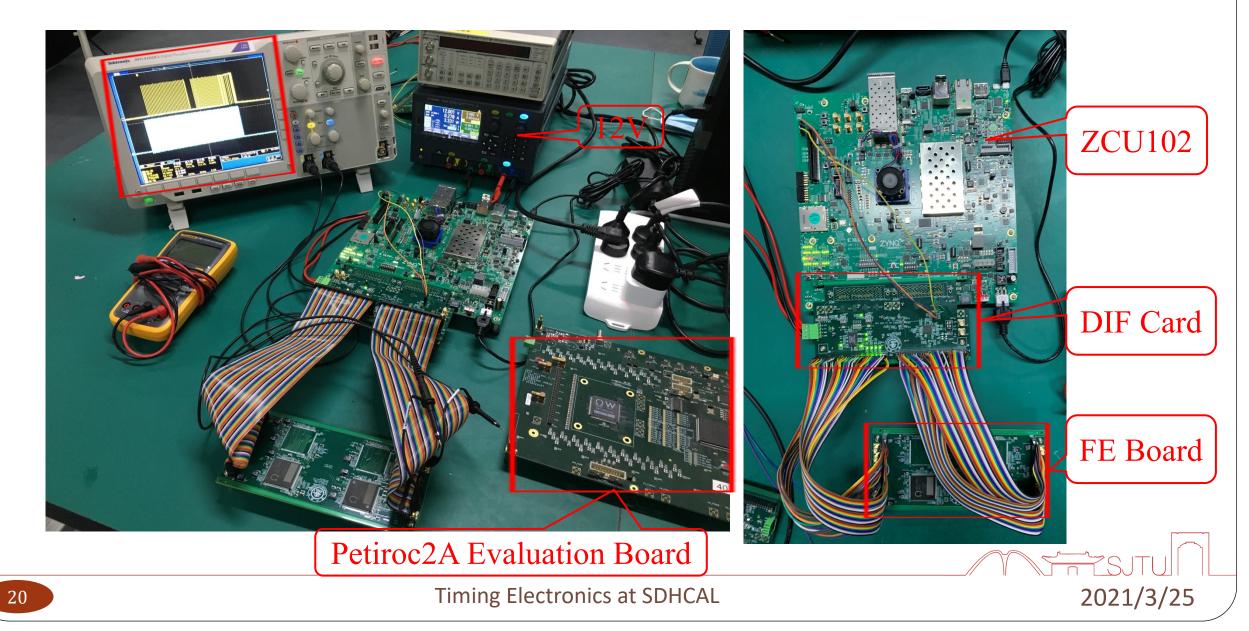


2021/3/25

- One channel split into two parts, respectively for charge and time measurement
- Internal DAC for each channel to adjust the amplitude of the input signal
- Lower power consumption (~6mW/channel)
- Jitter ~18 ps RMS on trigger output (4 photoelectrons injected)

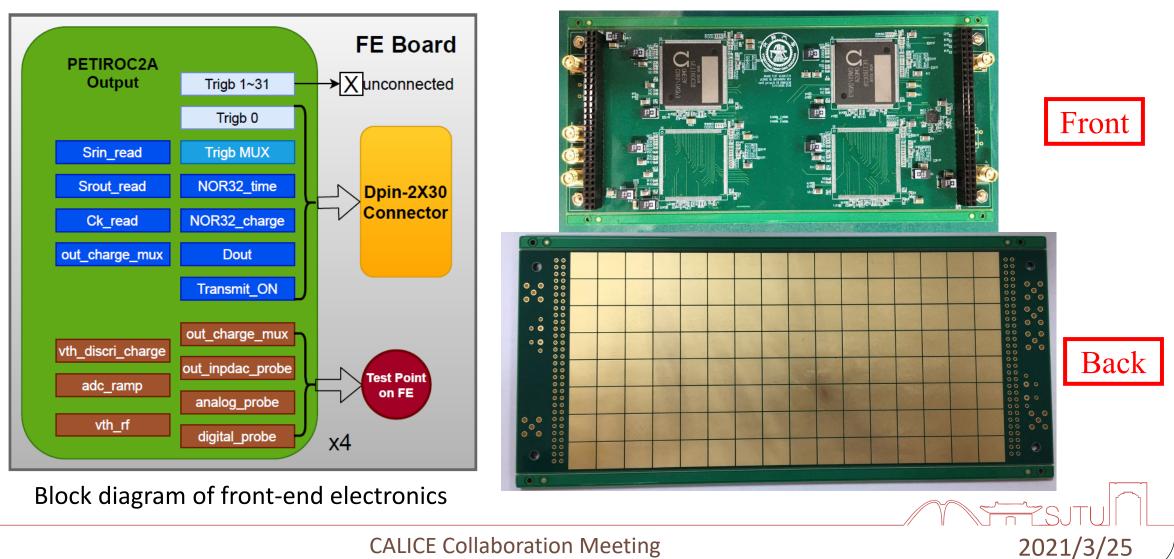


Test System and Setup



Sub-component Design and Testing

Front-End readout Board Design with pads and four petiroc2b

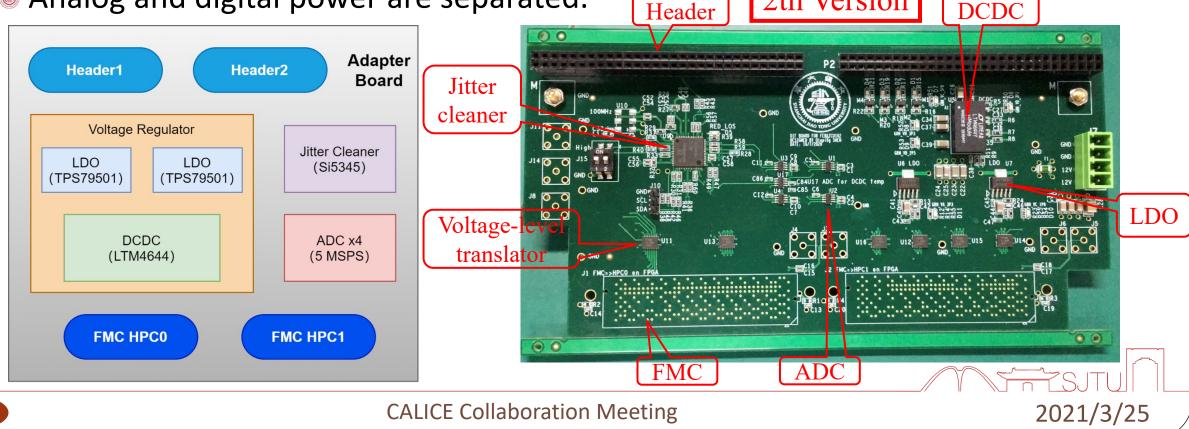


21

Sub-component Design and Testing

- Detector Interface Card Design: mainly jitter cleaner and power system
- IF card will be in charge of the communication and data transfer with the FE electronics(two headers) and ZCU102(two FMCs).
 More Details
- Analog and digital power are separated.

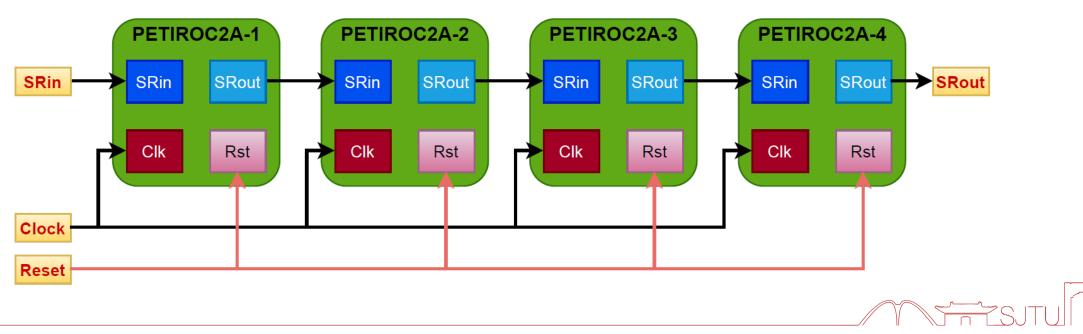
22



2th Version

Sub-component Design and Testing

- Front-End readout Board Design
- Somponents: 4 Petiroc2B, clock buffer, 2 headers, SMA for signal injection .
- I28 pads at the bottom, induction unit size: 1cm \times 1cm.
- The dimension:197mm*82mm, the blind/buried via technology.
- Sonfiguration for petiroc2a with daisy-chain (SPI sending shift register data)



2021/3/25

Details on DIF Design and Testing

Solution Strategy Strategy

