Hadron Universal Logic module User Guide

2023.08.19

Ryotaro Honda

Open-It HUL project

Table of contents

1. Introduction

1.1 Update history

2017.12.19

- In Chapter 2, added a section about Mezzanine DTL.
- In Chapter 2, added a section about Mezzanine HR-TDC.
- Consistent behavior of the reset sequence in the existing firmware (described in Chapter 4).
- Explicitly stated that 100 Mbps ethernet communication is not supported in the exisiting firmware.
- Added a bit in the data header of the existing firmware showing that HRM exists (described in Chapter 4).
- Stability improvement of HUL MH-TDC in high rates. Increased the event word size to 12 bits. Confirmed no event slips up to 10^9 events.
- Added HR-TDC section in Chapter 4.
- Added a few practical use in Chapter 7.
- Description about the board with SPI flash chip changed to MT25QL512.
- Description of simple tests.
- Description of how to use multiple HUL in a VME crate.
- Usage of HR-TDC.
- Modification to sitcp_controller.cc in controlling C++ software package. Sleep configulation in Reset_SiTCP.
- Removed gzfilter.hh from the controlling C++ software package, because it is strongly dependent to the gcc version. Removed data compression option.

2018.2.2

- Resolved an issue of returning a wrong event ID by +1 when event-tag is received from J0 bus by HRM, Scaler, MH-TDC and HRTDC_BASE, relative to the module with HRM installed.
- Resolved an issue of data returning out of the search window range from Mezzanine HR-TDC and MH-TDC.
- Resolved an issue of Local bus hang in case of BCT::Reset in HR-TDC_BASE.
- Described in Chapter 7 the issue that no triger could be received in the J0 slave modules after a module reset in case of Level2 trigger in J0 on a VME crate (not resolved).
- In Chapter 7, described the case where an event slip has occured.

2018.8.22

- In Chapter 7, described a board which has SPI flash upgraded to S25FL256SAGNFI001.
- In Chapter 7, described the handling necessary in the original SPI flash memory, N25Q128A

2021.11.10

- Firmware is major updated to version 3.
- Major updates of the descriptions in the User Guide

2022.01.13

- Release the English version user guide.

2022.03.18

- Correct the description of the mezzanine HR-TDC sub-header structure.

2023.01.17

- Released Mezzanine HR-TDC v5.0 and HUL HRTDC BASE v4.0. Updated description of what changed.
- Removed MifFunc.cc from software. Introduced BctBusBridge.cc instead.
- Fixed a bug that the self-busy length of HUL HRTDC BASE is short and there is a valley in the busy signal.

2023.02.24

- Released HUL HRTDC BASE v4.1. The bug-fixed version of v4.0, which does not work correctly.

2023.08.19

- Fixed incorrect description of Mezzanine HR-TDC clock frequency.

1.2 Module overview

Hadron Universal Logic (HUL) module has been developed as an upgrade from Tohoku Universal Logic (TUL) which was used in Hadron Hall experiments, J-PARC. The controlling field programmable gate array (FPGA) is upgraded to Xilinx Kintex 7, enabling more complicated logic conditions run in a higher speed. HUL has two fixed input connectors and two mezzanine slots the latter of which enable extensions to various experimental needs by installing mezzanine cards. These two connectors and slots are connected to FPGA by 64 (32x2) fixed input lines and 64 (32x2) pair differential lines, respectively, enabling handling of 128 channel maximum inputs. The 64 pair differential lines are directly connected to the mezzanine base connectors, so that input/output is programmable to the mezzanine cards. HUL is equipped with the data communication ethernet interface (GbE) and the trigger input/output busline (KEK-VME J0) which TUL did not have. Communication to the data-taking PC employs UDP and TCP protocols supplied by the SiTCP technology, which is a FPGA-based hardware implimentation of TCP/IP, supporting 1Gbps TCP communication via an ethernet cable (Cat.5 and above). UDP communication is extended by remote-bus control protocol (RBCP) of SiTCP, which supports the addressed access to the memory region of HUL in slow-controls. KEK-VME J0 bus is 8 pairs of triggering bus in M-LVDS signal levels. It consists of 7 pairs of transmission lines and one pair of output (BUSY). HUL may act as the bus controller or the slave receiver of the J0 bus. HUL is developped in an Open-It project "Hadron Universal Logic Module", employing the technological developments achieved in the Open-It consourtium. The interectual property and its usage is subject to the terms of the Open-It consourtium. [Open-It]

2. Hardware

2.1 HUL controller module

Hadron Universal Logic (HUL) controller module is VME 6U size printed circuit board, and is called simply as HUL hereafter. The catalog number in GND Ltd. is: **GND catalog number (ジーエヌディー管理番号) GN-1573-1**

2.1.1 Detailed Specifications

- Input ports
  - 64 pairs of differential inputs (KEL 8831E connector)
  - Supports LVDS, ECL, PECL, LVPECL etc.
  - 4 ports of NIM inputs
- Output ports
  - 4 ports of NIM outputs
- Mezzanine slots
  - 2 slots
  - Directly connected to FPGA via duplex LVDS 32 pairs
  - Power suppy to Mezzanine board: +3.3 V from HUL
- Communication interface
  - RJ45: GbE (1000BASE-T)
  - VME J1 is not supported
- FPGA
  - Xilinx Kintex7 (XC7K160T-1FBG676C)
- Configuration memory chips
  - SPI flash in 3.3V (one of the followings depending on the manufacture date)
    - N25Q128A13EF840E
    - MT25QL512ABB1EW9-0SIT
    - S25FL256SAGNFI001
  - SPI (synchronous serial interface) in serial configuration mode
- Clock source
  - 50 MHz LVCMOS (~50 ppm)
    - Peak-To-Peak jitter 30ps
- Trigger bus
  - KEK-VME J0
  - HUL may be configured as a bus driver or a receiver
- Power supply
  - DC +5V
  - supplied from a AC/DC adapter, or the VME-J1 connector
- Power consumption
  - static: 0.5~0.7 W @3.3 V (mainly at ethernet PHY) and 0.5~0.7 W @1.0 V (mainly at FPGA)
  - dynamic: strongly depends on the FPGA firmware

A picture of HUL and block diagram are shown below

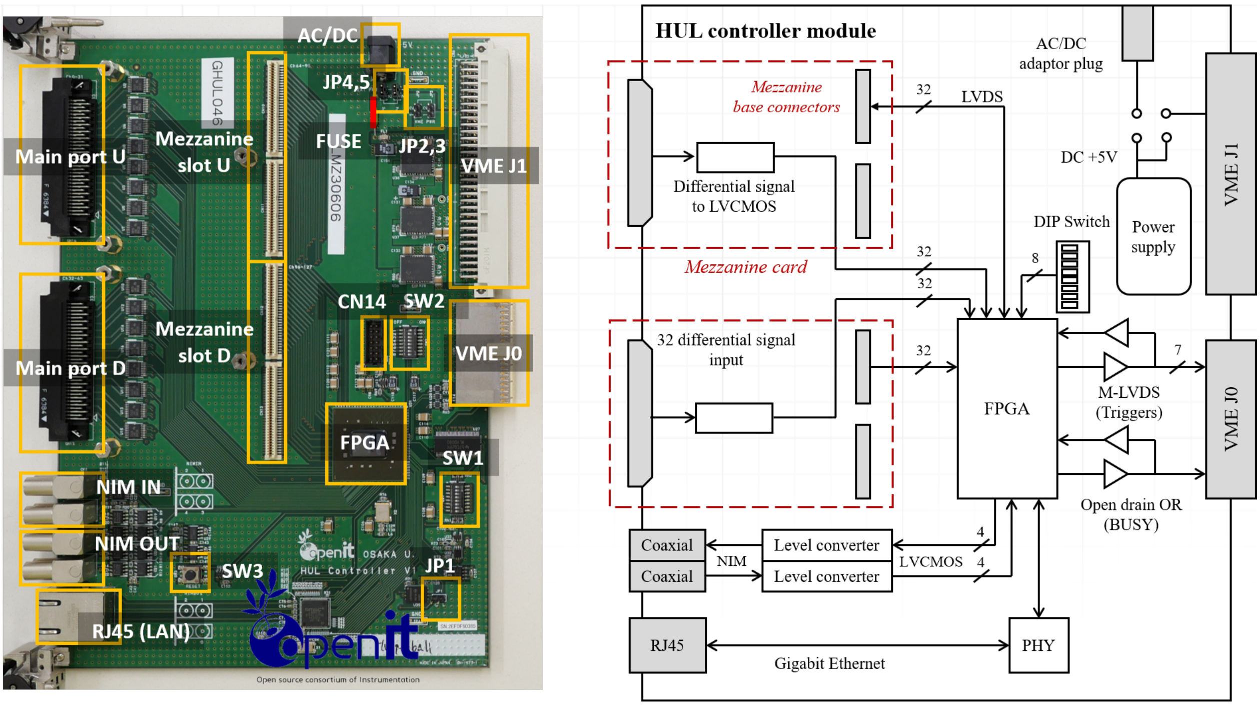

Figure 1: Photo and block diagram of HUL controller.

Main port U (D)

Fixed input ports in the front. The connector is the half-pitch 68 poles (KEL 8831E-068-170L-F). A cable with the compatible connector is supposed to be manufactured. Channel assignment is 0-31 for U and 32-63 for D, with lower number in the marker side. The signal grounds are A1A2 and B1B2 pairs, right below the marker. The inputs support differential signals in LVDS, ECL, PECL and LVPECL levels with the common mode voltage in the -4 V ~ +5 V range. When feeding an emitter-follower type signal levels, such as ECL, the signal current must be controlled in the driver side. There is no register for the current control in the HUL side.

The fixed input ports converts the differential signals to LVCMOS before feeding to FPGA. This limits the maximum repetation speed to 560 Mbps. If the signal rate is higher than that, consider using a mezzanine card. For an application to wire chambers, where Amp Shaper Discriminator (ASD) outputs are fed into HUL, the repetition rate limit is not a problem.

NIM IN

4 intputs of NIM level. Channel assignments are written on the board. **Channel number starts with 1**

NIM OUT

4 outputs in NIM level. Channel assignments are written on the board. **Channel number starts with 1**

Ethernet connection (LAN)

RJ45 type ethernet connector. Gigabit Ethernet PHY chip is connected. Used for PC-FPGA communication via SiTCP.

Mezzanine slot U(D)

Slots to install mezzanine cards. HUL has two of 64 pole connectors MOLEX-071439-0464 for one card. The upper connector has the signals and the lower has the power supply and the ground. Mezzanine cards may employ a few models of connectors with different heights, but if MOLEX-071436-1464 is employed, the card will be supported by three of 9 mm stand + 0.5 mm washer sets.

The signals are 32 pairs of differential lines to FPGA. There is no component to determine the signal direction between mezzanine connectors to FPGA, so that duplex LVDS may be adopted for the signal levels. The design of the mezzanine card and the FPGA firmware determine the actual signal levels and directions. If inputs are assigned to Slot U and D, the channel numbers will be 64-95 and 96-127 respectively, with the offset (0-31 and 32-63) from the main ports of the board. Some of the polarities p/n are reversed between the mezzanine connectors to FPGA owing to the simplicity of the pattern layout. The VHDL source `MZN_NetAssign.vhd` takes care of the polarities of the differential signals by inserting logical inverters (NOTs) to the signals.

Mezzanine slots supply +3.3 V to the card. The current allowance will be 4-5 Amps (13-16 W) to the card, with the 6 Amps supplied from the power and 1-2 Amps consumed by HUL itself.

CN14

Connector for JTAG protocol. PC may download the FPGA firmware via USB-JTAG downloader by Xilinx. How to download bitstream or MCS are described more in detail in Chapter 7.

SW1

The switch which defines whether HUL is a VME J0 bus driver or a receiver. SW1-8 must be all OFF to be a receiver (default). SW1-8 must be all ON to be a driver. Do not define more than one driver in a VME crate, as such situation will short circuit the bus line and causes a damage.

SW2

User defined dip switch. The role is defined in FPGA firmware.

SW3

User defined push switch. It generates a pulse to send to FPGA. In the exisitng firmware, it causes the highest reset action. The pulse logic is negative, changing from 1 to 0 if pressed.

JP1

Reserved. Keep it open.

JP2, 3

Close to take power from VME J1. DC +5V as the supply.

JP4, 5

Close to take power from a AC/DC adapter.

AC/DC

Standard socket for a DC power input (OD 5.5 mm, ID 2.1 mm, center plus).

FUSE

5 Amps fuse. Compatible to a standard fuse, such as PICO® fuse by Littelfuse.

VME J1

Connector to a VME crate. Only the power supply (+5V) is connected.

VME J0

Connector to KEK-VME J0 bus. There is no connection to the power supply on J0 bus. **Equipped as a default, but is an optional.** Order the board without CN9 to remove this connector.

2.2 HUL Mezzanine cards

2.2.1 HUL Mezzanine Drift Chamber Receiver (DCR) Ver.1

Drift Chamber Receiver (DCR) Ver.1 is the input repeater circuit board for the Amp Shaper Discriminator (ASD) card designed for drift chambers (GND catalog number: GNA200). It repeats 32 channels of signal from GNA200 into LVDS levels, so that HUL can read. Ver.1 keeps the differential characteristics to FPGA with a higher timing resolution, it is superior to the Ver.2 or the HUL main input ports. However, these advantages are subtle, and Ver.2 is upper conpatible with more kinds of supported signal levels.

GND catalog number (ジーエヌディー管理番号): GN-1573-S1

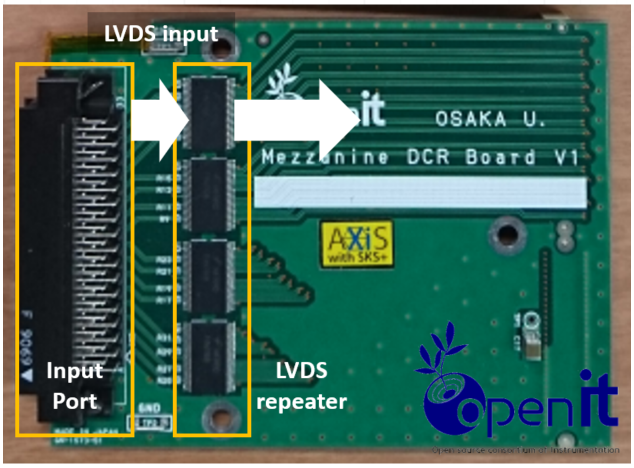

Figure 2: mezzanine DCR v1

Input Port

LVDS input ports. The connector is a half-pitch 68 pole (KEL 8831E-068-170L-F). A compatible connector must be used with cables. The marker side is lower in the channel number, as in the HUL on-board inputs. The grounds are A1A2 and B1B2 pair, directory below the marker. Due to the simplicity of the board pattern, DCR ver.1 and ver.2 both have a swapping of the channel numbers and polarity p/n. The VHDL source `DCR_NetAssign.vhd` takes care of the channel / polarity order to be correct, as shown in the figure below.

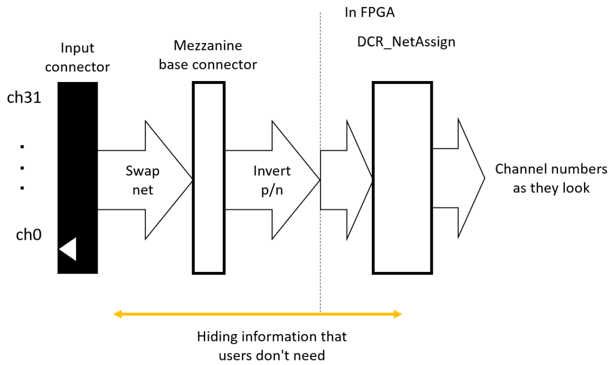

Figure 3: DCR has net swaps and corrections

LVDS repeater

Receives LVDS signals and repeats to FPGA. These ICs protect FPGA from unintentinal signal disturbances, such as discharges.

2.2.2 HUL Mezzanine Drift Chamber Receiver (DCR) Ver.2

Drift Chamber Receiver (DCR) Ver.2 has the same role with ver.1, but employs the same differential receiver ICs with those used in HUL on-board main input ports. This enables acceptance of various levels of differential signals (LVDS, ECL, PECL, LVPECL etc) as for the HUL on-board inputs. For the purpose of input ports extention, ver.2 is recommended. **GND catalog number (ジーエヌディー管理番号): GN-1626-1**

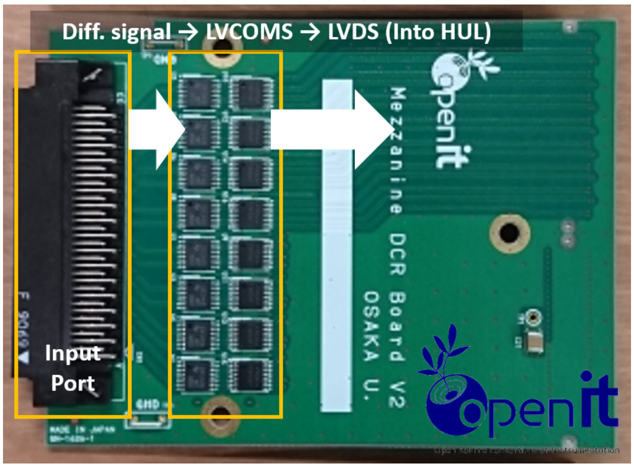

Figure 4: mezzanine DCR v2

Input Port

Mechanically the same with DCR Ver.1. Supports differential signals with the common mode voltages ranging from -4 V to +5 V. The differential signal is converted to LVCMOS (as in the on-board input ports) and subsequently converted to LVDS to feed to FPGA. The VHDL code DCR_NetAssign.vhd takes care of the channel and polarity p/n swaps as in ver. 1.

2.2.3 HUL Receiver Module (HRM)

HUL Receiver Module (HRM) receives the trigger signals and event numbers distributed from Master Triger Module (MTM: GNN-570) employed in J-PARC Hadron Hall experiments, and returns BUSY signals back to MTM. HRM de-serializes the trigger signals and event numbers transmitted through the Category 5e twist-pair cables, and converts to a bus signal format to FPGA. In return, HRM serializes the BUSY and RSV2 signals from FPGA to be transmitted to MTM. The trigger signals to and BUSY signals from HRM are all processed in the FPGA on HUL, enabling the HUL to function as a J0 bus controller of KEK-VME crates.
GND catalog number (ジーエヌディー管理番号): GN-1627-1

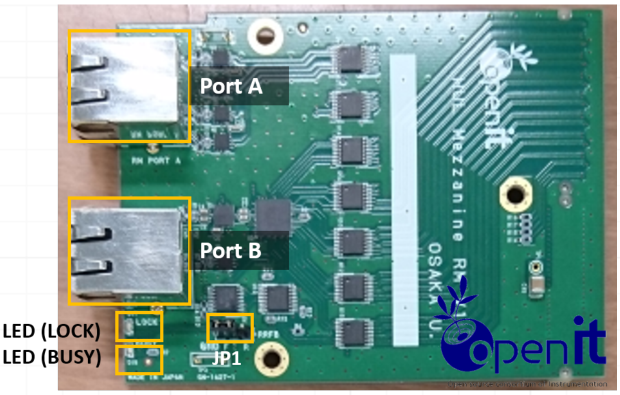

Figure 5: HUL receiver module (HRM)

Port A

Trigger inpurt port A. Should be connected with a twist pair cable to MTM port A or repeater port A.

Port B

Trigger input port B. Should be connected with a twist pair cable to MTM port B or repeater port B.

LED (LOCK)

Lights green if the event tag bit from Port B is correctly decoded, and the PLL lock of the clock is high.

LED (BUSY)

Lights red if the BUSY to MTM is high. However, HRM reflects only the BUSY status from FPGA, which should be independently checked if it correctly reflect the the create bus BUSY signals.

JP1

Controls RRFB, meaning whether the de-serialized event tag outputs synchlonize to RCLK clock rising edge (R) or falling edge (F). Default is R. This is a 3-pins pin header. Short the R side.

Inputs/Outputs to/from Mezzanine HRM

Signal	Direction	Description
RCLK	IN	Clock decoded from the event tag from port B
LEVEL1	IN	Level1 trigger signal
LEVEL2	IN	Level2 trigger signal
Clear	IN	Fast clear signal
RSV1	IN	Reserve 1 signal from MTM
LOCK	IN	Lock bit of PLL, showing that the clock is correctly decoded.
SNINC	IN	Spill Number Increment. FPGA may count the spill numbers.
Event counter	IN	12 bit event number from MTM
Spill counter	IN	8 bit spill number from MTM
RRFB	OUT	Timing for decoded tag information. High for rising edge (R) and Low for falling edge (F) of RCLK.
BUSY	OUT	BUSY signal to MTM
RSV2	OUT	Reserve 2 signal to MTM

2.2.4 HUL Mezzanine differential signal transmitter LVDS (DTL)

Mezzanine DTL is a LVDS output buffer from HUL-FPGA to other devices. The circuit is the same with DCRv1, except the input/output directions of the LVDS repeaters. The channel assignment is the same with DCRv1. HUL-FPGA must output LVDS signal to drive this card. **GND catalog number (ジーエヌディー管理番号): GN-1724-1**

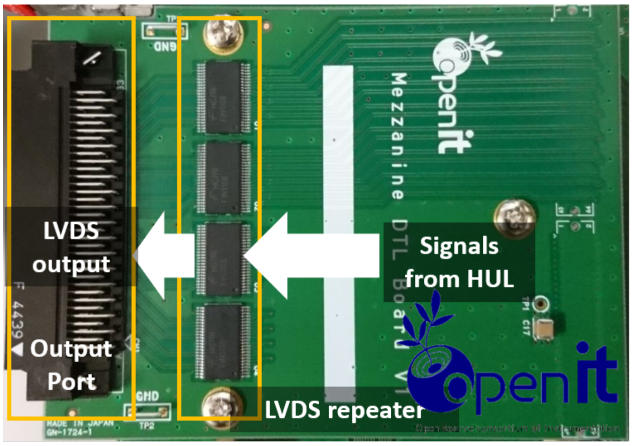

Figure 6: Mezzanine DTL

Output Port

LVDS output ports. The mechanical specification, signal and ground assignments are the same with DCR v1. This mezzanine card requires to use the VHDL source `DCR_NetAssign.vhd`, but the input/output direction is opposite from the case in DCRv1.

2.2.5 HUL Mezzanine NIM extension (NIM-Ex)

Mezzanine NIM-Ex extends the NIM inputs/outputs of HUL. It has 12 LEMO connectors grouped in 6 pairs which are independently selected for inputs or outputs by switch settings. The same IC is employed as used in HUL NIM IO ports, and the speed is the same.

NIM-Ex card consumes 5.5W, being rather large. The card generates -3.3 V on-board and may become hot. Air cooling fan is strongly recommended. Otherwise, the mezzanine card may break by the self-heating. There are numerous mistakes on the circuit diagrams. An example design is included in the skeleton project of mezzanine board, because the errorous circuit net prevents a guess of correct XDC and HDL code.

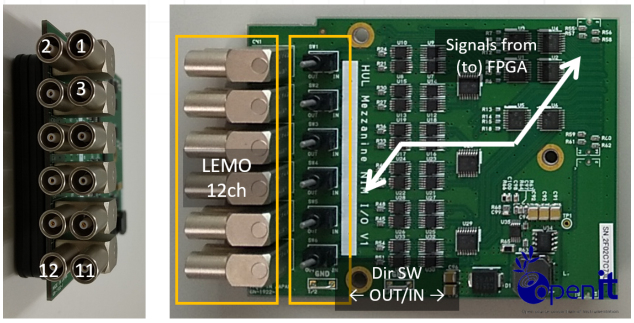

Figure 7: Mezzanine NIM-Ex

LEMO

12 channel LEMO connectors. Example sckeleton design includes `NimEx_NetAssign.vhd`, which has the net number associated to the physical channels as figure above.

Dir SW

Signal direction in groups of 2 channels. Defined for OUTPUTs if the switch is toward the LEMO connector; defined INPUTS if otherwise.

2.2.6 HUL Mezzanine High-resolution TDC

Mezzanine HR-TDC is capable of measuring time difference in ~30 ps resolution in common stop and multi-hit mode. It has a dedicated FPGA for the time measurement on the mezzanine card, and requires controls and data transfer operations from HUL. The function of the mezzanine is to measure time, to form data in a pre-fixed format and transfer the data to FPGA on HUL by the trigger input. Depending on the firmware of FPGA on HUL, this mezzanine is able to act as a simple TDC or play a more complicated role such as Time of Flight (TOF) trigger. Detailed action and control is described in Chapter 4.

The FPGA on this mezzanine card is Kintex-7 160T-1, the same chip as used on HUL. One mezzanine takes 32 channels of timing signals, capable to measure both the leading and trailing edges. The sinal input level is exclusively LVDS, not supporting ECL. The clock for FPGA is selectable either from the quartz on the mezzanine or from HUL. The power is +3.3 V taken from HUL, and other voltages are generated on the mezzanine card via low drop-out (LDO) regulators. The power consumption is 5 W / card, which is rather high. If the +3.3 V power supply chip on HUL is LMZ30604RKGT (max. 4 Amps), it is insufficient to drive two mezzanine cards. In such cases, make sure the chip on HUL is **LMZ30606RKGT (max. 6 Amps)**. The power consuption of the entire board will become almost 20 W when two HR-TDC mezzanine boards are installed. Make sure that the board is well cooled by a fan. If the cooling of the FPGAs is not sufficient, they became unstable with progressively increased leak currents. It is strongly recommended that a VME crate with a fan is used. (Also, stable ground is necessary to reach to the high timing resolution.) **GND catalog number (ジーエヌディー管理番号) : GN-1644-1**

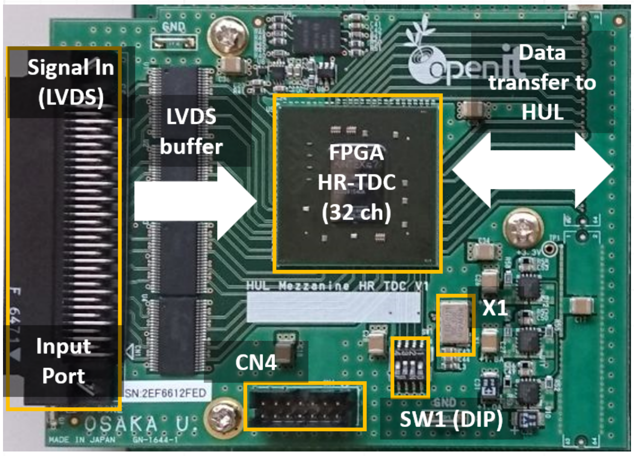

Figure 8: Mezzanine HR-TDC

Input Port

Signal inputs which supports only LVDS levels. Connector is a half-pitch 68 pole (KEL 8831E-068-170L-F). The ground is assigned on A1A2 and B1B2, right below the marker. Channel assignment is 0-31 from the marker. There is no swap of the channels.

CN4

JTAG connector for firmware download. FPGA and SPI flash memory may be accessed. The SPI flash memory chp is N25Q128A11EF840E or MT25QU256ABA1EW9-0SIT with 1.8V power inputs. Please configure / identify the chip correctly when downloading MCS onto the SPI chip.

SW1

Dip switch to control the FPGA of the mezzanine card. The current firmware selects the clock source from the quartz on the mezzanine card or from the HUL.

X1

100 MHz quartz oscillator. The frequency must be the same if HUL clock is used.

SPI flash memory chip version

Mezzanine HR-TDC has two versions for the flash memory chip, depending on the date of the manufacture. To correctly identify the flash memory chip, Vivado (the FPGA tool on PC) is required. It is safer to mark the SPI flash memory chip version with a note on the board.

- N25Q128A11EF840E (pre 2020 manufacture)
- MT25QU256ABA1EW9-0SIT (since 2020)

3. SiTCP

SiTCP is a hardware implementation of TCP/IP network communication protocol, developed by Tomohisa Uchida at KEK. It enables the communication without an involvement of a CPU. Please refer to Uchida's and BBT's web sites for detailed usage and the most recent IP core (the component to be linked when developing a firmware). * Uchida's web site * BBT's web site

SiTCP block diagram is shown in a figure below. SiTCP initializes the ethernet communication PHY and loads MAC and IP addresses from EEPROM. In the forced default mode, the MAC and IP addresses are set to the default (192.168.10.16). The forced default mode does not allow other SiTCP hardware in the same forced default mode to exist in the same network, usually used in the 1 to 1 connection to the PC for testing. SiTCP provides TCP and UDP communication protocols to FPGA. SiTCP supports both the 100 Mbps and 1Gbps modes automatically switched by the signal from PHY; however, the existing firmware for HUL only supports the 1Gbps mode. TCP generally transmits or receives data in the 8 bit units synchlonized with the system clock; however, the TCP receive action (data from PC to hardware in TCP) is not recommended in SiTCP. The registers on the hardware are accessed via UDP. A unique packet called RBCP is used in the UDP for control command transmission, address setting and data transmission and receiving. RBCP returns UDP acknowledge for handshaking between the PC and SiTCP. UDP transmits 32 bit address and transmits/receives data in 8 bits synchlonized with the system clock. SiTCP internal registers and the EEPROM are also addressed and may be accessed via UDP, however do not touch them unless really necessary. The IP and MAC address are stored in EEPROM and loaded to SiTCP registers, which may be accessed by the BBT software (SiTCP tools) or specifying the UDP address in a user program.

SiTCP does not "keep alive" the connection by default. This causes a session closure in case of very low trigger rate (~1 trigger / 30 min). To avoide such situation, regularly send a keep alive packet.

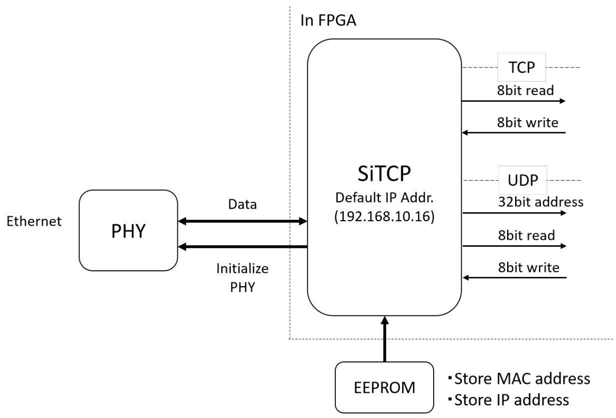

Figure 1: SiTCP block diagram

3.1 MAC and IP address setting

MAC and IP addresses are stored in EEPROM and designed to be set via SiTCP. A firmware which has at least SiTCP must be loaded to FPGA to access the address. To set the address, a user program with RBCP communications may be written, however, SiTCP tools developed by BBT is also available from its web site after user registration. To write the MAC address, use SiTCPMpcWriter in the BBT's tools. Select the file in the DVD purchased from GND Ltd (recommended) after starting the tool. MAC address has to be unique to each hardware; having the same MAC address in the same network causes a communication trouble. For more about SiTCPMpcWriter, refer to BBT's web page.

To write IP address, use SiTCP Utility (MS Window software) also available in multi platforms (beta-release). This procedure must be performed after installing the MAC address. Check on "access to EEPROM" which is marked in the red square in the figure below. Specify the current IP address and press "show(表示)" button. If the "eye" in the upper right corner blinks and the information is shown, the access was successful. If the current IP address is unknown, start the hardware in the foced default mode (192.168.10.16). Fill in the new IP address to set and press "rewrite(書き替え)". This loads the new IP address only to EEPROM, and the current IP address stays the same. To confirm the IP address in EEPROM, again press the "show(表示)". This address is effective after turning on the harware power next time

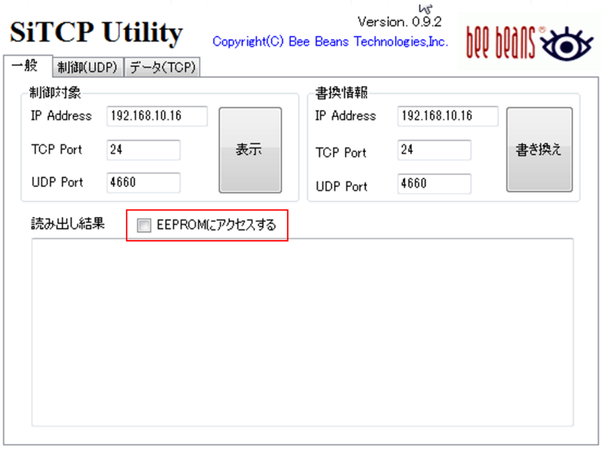

Figure 2: SiTCP Utility

4. Firmware

4.1 Version.1 and Version. 3 series

Major update of the firmware was done in August 2021 from version 1 to version 3. There is no software compatibility between them, because of the change of the local bus controller (BCT) structure. Version. 2 is an internal development and not released. Version.1 uses some of the `RBCP address` as the data bus, but version.3 uses the full `RBCP address` (32bit) and the full `RBCP data` (8 bit) as they are. The modification above seamlessly enables `FPGAModule` class as a wrapper to `RBCP` class. In the DAQ / slowcontrol software for version 3 firmware, `RBCP address` specifies the local modules, making `ModuleID` and `LocalAddress` in version 1 obsolete. `ModuleID` and `LocalAddress` reflects the Bus Controller (BCT) structure of version 1, which information is irrelevant to the (non developper) users.

Version.3 is newly equipped with two local modules common to all: Flash Memory Programmer (FMP) and Self Diagnosis System (SDS). FMP writes SPI flash memory via SiTCP. It has enabled a remote downloading of the firmware (MCS file) to EEPROM via ethernet, eliminating the offline download procedure using a USB-JTAG cable. SDS diagnoses the status of FPGA and the board. It audits the number of soft error corrections caused by radiations, or detects an un-recoverable error. Surveillance of FPGA temperatures and critical voltages (VCCINT, VCCAUX and VCCBRAM) are also implemented.

4.2 Basic structure of HUL firmware

There are two types of HUL firmware: with or without data acquisition by the PC. An example for the former is TDC, and the latter is a trigger logic. The latter has the individual block structure depending on the function, but the former (data-acquisition) has a common block structure. The following is the common block diagram. The components shown in orange in the figure are the functions implemented in all firmware. The `local bus` that controls each local module is controlled by the local bus controller (BCT), which is connected to SiTCP's RBCP (UDP). FMP and SDS are local modules implemented in all version.3 firmware, and their functions are as described above.

The white squares in the figure represent the parts related to data acquisition. They are called a measurement block, such as TDC and scaler. These features are mainly implemented in the block labeled as *User circuit*. The data from these *user circuits* is collected in the event builder block via the builder bus, attached with event headers, and then transferred to the PC via SiTCP. The Receiver block, trigger manager, and J0 interface are the blocks involved in trigger I / O. HUL is designed to receive trigger and event tag information sent by Master Triger Module (MTM: GNN-570). The receiver block is activated when the HRM mezzanine card is installed, and gains the role of receiving the trigger from the upstream circuit. This block is also connected to the builder bus to transfer the received trigger information to the PC when the trigger is received. When using the receiver block, HUL is most likely the J0 bus master in the VME crate. The J0 interface sends the trigger and tag information received by the receiver block to the J0 bus if the HUL is the J0 bus master. In the case of HUL being a J0 slave, the trigger information coming from the J0 bus is passed to the trigger manager. To make HUL a general-purpose module, it supports trigger input from the NIM port in the absence of MTM. The trigger manager is responsible for trigger distribution inside the firmware, and the trigger input source may be selected from the HRM mezzanine (receiver block), NIM input, and J0 bus input. The *LEVEL2*, *Fast clear*, and *TAG* information sent by the Trigger manager is used internally by the event builder to generate event header information and determine the data transmission to PC.

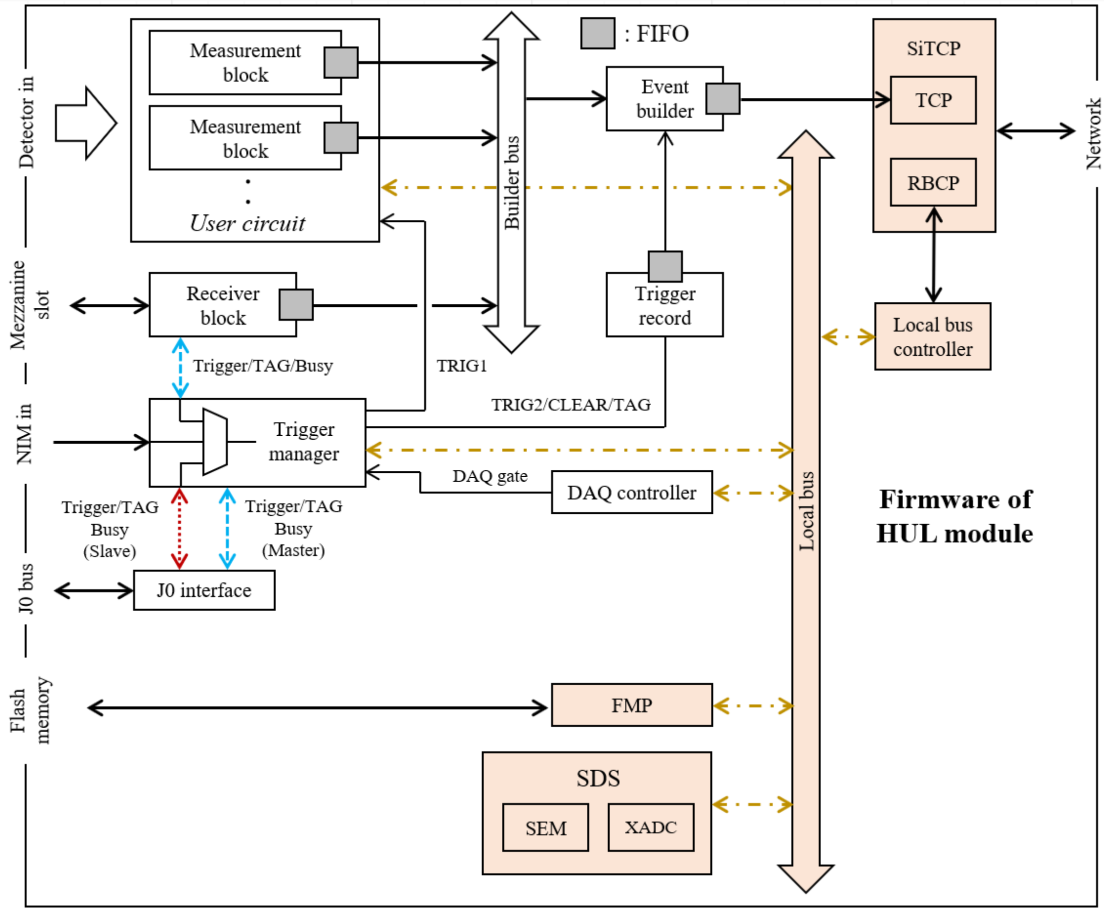

Figure 1: Block diagram of data-acquisition-type firmware

4.3 Commons to all HUL firmware

hul_software package manages the addresses of BCT, FMP, and SDS, the common parts of the firmware, in the file `common/src/RegisterMapCommon.hh`.

4.3.1 Reset sequence

From the update of December 19, 2017, the reset sequence of all firmware has been standardized. In the previous firmware, there was no way of resetting other than turning off the power, when the BCT hangs. In MH-TDC, some FIFOs were not reset even if BCT reset was applied, and the DAQ hang state was not cleared. Therefore, a few reset levels have been introduced to deal with the problem.

Reset procedures (from milder to harder)

- BCT reset: Writing BCT Reset command in the designated address with RBCP. User modules under BCT control are reset in the firmware signal. Normally used.
- SiTCP reset: Calling `Reset_SiTCP` defined in `sitcp_controller.cc`. Will be reset even if the BCT is hung or deadlocked.
- Hard reset: Pressing SW3 on the board. All circuits including SiTCP are reset. Most enforceable reset procedure of all.

Use BCT reset normally, and if BCT deadlocks by a mis-operation, use SiTCP reset. If SiTCP hangs as well, use Hard reset.

4.3.2 Network speed supported

The currently released firmware, SiTCP only works in the 1 GbE mode. SiTCP supports both 100 Mbps and 1 GbE modes and can be configured to automatically switch. However, the existing firmware disables the automatic switching function for the convenience of the PHY chip being used. 100 Mbps mode will never be implemented, because there would be no needs for it anymore. Do not connect to a 100Mbps-only network switch. SiTCP will not communicate.

4.3.3 Bus controller (BCT)

Bus controller (BCT) has the functions of issuing a BCT reset, acquiring a version numer, and reconfiguring the FPGA, in addition to the usual functions to access the local modules. The special functions are available by executing the following operations to the listed RBCP address.

RBCP address for special functions of BCT (Module ID: 0xE)

Register	address	operation	bit width	memo
Reset	0xE000'0000	W	-	Asserting module reset signal to all modules except SiTCP.
Version	0xE010'0000	R	32	Returns Firmware ID and version number. Needs to read 4 bytes.
Reconfig	0xE020'0000	W	-	Sends Low to PROG_B_ON to re-configure FPGA. SiTCP connection will be immediately closed, and may be reconnected in a few seconds.

4.3.4 Flash Memory Programmer (FMP)

FMP sends SPI commands from the PC and execute supported functions by the memory chip, such as read / write page data. Such SPI commands include erase, write and read memory. The same operations which Vivado does to the memory (erase, write, and verify) has now become available over the network by FMP. There is a known bug in FMP: the next page write request is accepted before the precious command finishes. Currently, the software simply waits a sufficient amount of time before sending the next page write command, but this is not a very good solution. Eventually, the FMP module may be corrected. Because of this wait time, it takes longer time to write than the network speed anticipates. If write-failures occur frequently, please contact the author of this document. The RBCP address is listed below; it is not recommended to control FMP from anything other than `FlashMemoryProgrammer.cc` included in the *hul_software* package. If the write-lock bit is accidentally set, that memory block may never be writitten again.

RBCP address of FMP (Module ID: 0xD)

Register	address	operation	bit width	memo
Status	0xD000'0000	R	8	Obtain the status bit of FMP module. Currently, the lowest bit is assigned to SPI command cycle busy.
Status	0xD000'0000	R/W	8	Mode change of SPI sequence. bit 1-2: SPI sequence mode • 0x0: Read mode • 0x1: Write mode • 0x2: Instruction mode bit 3: Dummy mode Sets chip select to be OFF, making the flash memory immune to the SPI sequence.
Register	0xD020'0000	R/W	64	SPI command
InstLength	0xD030'0000	R/W	3	SPI command length
ReadLength	0xD040'0000	R/W	10	page read length
WriteLength	0xD050'0000	R/W	10	page write length
ReadCountFIFO	0xD060'0000	R	10	read count for FIFO where the page-read data are stored.
ReadFIFO	0xD070'0000	R	8	8-bit (byte) wide readout from the read FIFO.
WriteCountFIFO	0xD080'0000	R	10	write count for FIFO where the page-write data are stored.
WriteFIFO	0xD090'0000	W	8	8-bit (byte) wide writein to the write FIFO.
Execute	0xD100'0000	W	-	execute the SPI sequence.

4.3.5 Self Diagnosis System (SDS)

SDS is a self-diagnosis program. Soft Error Mitigation (SEM) and XADC, IP cores of Xilinx FPGA, are implemented and monitored. SEM is an IP core that detects, corrects, and classifies single event upsets (SEUs). SDS detects the number of errors corrected and the occurrence of un-correctable errors. If the system falls into an uncorrectable state, reconfiguring the FPGA from the flash memory or performing a power cycle is necessary. Also, it is possible to intentionally inject SEU, but it will be an advanced use of SDS; please check how to use SEM in the Xilinx User Guide.

XADC is the collections of built-in ADCs in Xilinx FPGA HUL obtains FPGA temperature, VCCINT, VCCAUX, VCCBRAM voltages via XADC.

SEM has an unresolved issue. The incorrectable error staus may become 1 after the power is turned on in some FPGAs. The cause is not clear, but in such cases, reset the SEM once by executing `reset_sem` in the *hul_software* package after turning on the power.

RBCP address of SDS (Module ID: 0xC)

Register	address	operation	bit width	memo
SdsStatus	0xC000'0000	R	8	Obtain the status bit of SDS module.
XadcDrpMode	0xC010'0000	R/W	1	Select DRP mode of XADC. • 0x0: Read mode • 0x1: Write mode
XadcDrpAddr	0xC020'0000	R/W	7	Supply DRP address of XADC.
XadcDrpDin	0xC030'0000	R/W	16	Supply DRP data for input to XADC.
XadcDrpDout	0xC040'0000	R	16	Obtain DRP data from XADC.
XadcExecute	0xC0F0'0000	W	-	Execute DRP access to XADC.
SemCorCount	0xC100'0000	R	16	Number of corrections to SEU by SEM.
SemRstCorCount	0xC200'0000	W	-	Reset SemCorCount.
SemErroAddr	0xC300'0000	W	40	Supply address to \`inject_address\` port of SEM.
SemErroStrobe	0xC400'0000	W	-	Send a pulse to \`inject_strobe\` port of SEM.

Contents of SdsStatus

bit number	Status	memo
1	Over temp	Indicates that the FPGA temperature has exceeded 125 ˚C. Turn off the power of HUL immediately to cool down, because there is a serious lack of cooling.
2	Temp alarm	Indicates that the FPGA temperature has exceeded 85 ˚C. The cooling capacity is most likely insufficient.
3	VCCINT alarm	It indicates that the voltage of VCCINT has exceeded the normal range (0.97-1.03V). Some trouble is occurring on the board.
4	Reserved	
5	Watchdog alarm	Indicates that the SEM heartbeat signal is absent. Some trouble is occurring in the SEM.
6	Uncorrectable error	Indicates that the SEM has detected an uncorrectable radiation error. FPGA needs to be reconfigured.
7	Reserved	
8	Reserved	

4.4 HUL Skeleton

This project is the minimum configuration of firmware that implements SiTCP but does almost nothing. Please use it as a sample when making firmware for HUL. There are only two functions, one is to take OR of the input signals and output it to NIM, and the other is to illuminate the LED with SiTCP or DIP. For the input signals, the fixed inputs (main input ports) and mezzanine inputs (assuming DCR v1 or v2) are grouped in 32 channels of one connector and their OR signal appears in the four NIM OUTs. Skeleton is misspelled in the VHDL source, but left as it is, because a correction influences in a wide range of codes.

An example for using the NIM-Ex mezzanine card is included under `sources_1 / example_design /`. Please refer to the toplevel.vhd enclosed.

Firmware ID and the current version

When Version is read from BCT, a 32-bit register is returned. Of these, the upper 16 bits are the Firmware ID, and the lower 16 bits are the version numbers (major version 8bit + minor version 8bit). The current HUL Skeleton ID and versions are as follows:

Firmware ID	0x0000
Major version	0x03
Minor version	0x02

In the following, versions are written in a format like v3.2.

Version history

Version	Release date	Modifications
-	-	There is no versions up to v2.x
v3.2	2021.08.01	Updated to Version.3

4.4.1 Register map

RBCP address of LED (Module ID: 0x0)

LED assignment

LED1-3 reflects LED register or bit 2-4 of SW2. LED4 is connected to "Over temp" flag of SDS.

4.5 HUL RM

By mounting the Mezzanine HRM, the HUL RM can be a J0 bus master and can operate as a DAQ module that reads the data received by the HRM. It is the basis of the MH-TDC and scaler firmware, and should be used as a starting point when developing a new DAQ type firmware.

Since the response to the trigger input is based on this firmware, here we will explain in detail the trigger system and the response of the event builder to it.

Firmware ID and the current version

Firmware ID	0x0415
Major version	0x03
Minor version	0x02

Version history

Version	Release date	Modifications
v1.0	2016.12.23	Initial release
v1.1	2017.01.15	RVM data header changed from 0x9C to 0xF9.
v1.2	2017.01.27	Vivado update 2016.2 => 2016.4 TRM middle buffer changed from disperse RAM to BRAM. The depth changed from 128 to 256. Prog Full threshold introduced. The reason for the depth of 256 is to be below SCR block depth. RVM middle buffer depth is also changed to 256. Functionality seen from the outside of the module stays the same.
v1.3	2017.03.22	Solved the problem that the initial register of IOM was not set correctly. Solved the problem that the number of words written in header 2 became 0 in the first event after turning on the power.
v1.4	2017.05.09	Solved the problem of not responding to Clear (BUSY stays standing).
v1.5		Solved the problem that HRM hangs when Clear is entered. (Replaced by v1.6 without release.)
v1.6	2017.08.22	Fixed an issue where DAQ would hang if a hard reset was entered within ~2 us after the trigger. A new register for each block to select its response to the hard reset. Added new local address.
v1.7	2017.12.19	Standardlized reset sequence. Bit 24 of Header3 is now indicating whether HRM exists (to be exact, whether DIP2 is ON).
v1.8	2018.02.02	Solved the bug that the event tag coming from the J0 bus was latched too early and the event number on the HRM side deviated by 1.
v2.x	-	un released
v3.2	2021.08.01	Added FMP and SDS. Installed Builder bus. Changed the structure of Local bus.

Overview of module function

HUL RM data acquisition block diagram is shown below. The modules which do not relate to data acquisision are omitted. The function of HUL RM is the processing of information received and decoded by Mezzanine HRM. Therefore, the function is **implemented only in mezzanine slot U**. The information received by Mezzanine HRM is distributed in three routes. The first is the Receiver Module (RVM), which stores lock bit, spill number increment, spill number (full 8bit), and event number (full 12bit) information and passes them to the EVent Builder (EVB). Data is passed to the Event builder via the builder bus, so the RVM information is contained in the data body. That is, the RVM is part of the measurement block for the EVB. The second route is the distribution of trigger information to the J0 bus. At this stage, the event number is reduced to 3-bit and the spill number is reduced to 1-bit. The third is the input to TRigger Manager (TRM).

The HUL firmware has a module called trigger manager (TRM) that manages the internal trigger distribution. TRM receives trigger signals from three ports, trig Ext (NIM IN), J0 bus (if slave), and HRM (if any), and the register sets which port to receive the trigger.

TRM distributes level1 trigger, level2 trigger, clear, and event number (3-bit) and spill number (1-bit) information to each measurement block and EVB. For EVB, these are not DAQ data, so the tags from TRM are embedded within the event hearder. The event number and spill number distributed by TRM have been reduced to 3-bit and 1-bit, because this information must be independent to whether HUL is a master or a slave to the J0 bus. If Tag is not received, both event number and spill number will be 0.

I/O Manager (IOM) is a module that controls which FPGA internal signal is assigned to the NIM input / output ports.

There three types of busy signals. First is the module busy, whici is a logical sum of internal blocks in firmware. Second is the J0 bus busy received from the J0 bus; this is a logical sum of busy signals from HUL, which are slave to the J0 bus. Last is the crate busy; this is a logical sum of the module busy, the J0 bus busy, and the external busy received from NIM-IN. We use the J0 busy and the crate busy, when HUL is a master to the J0 bus.

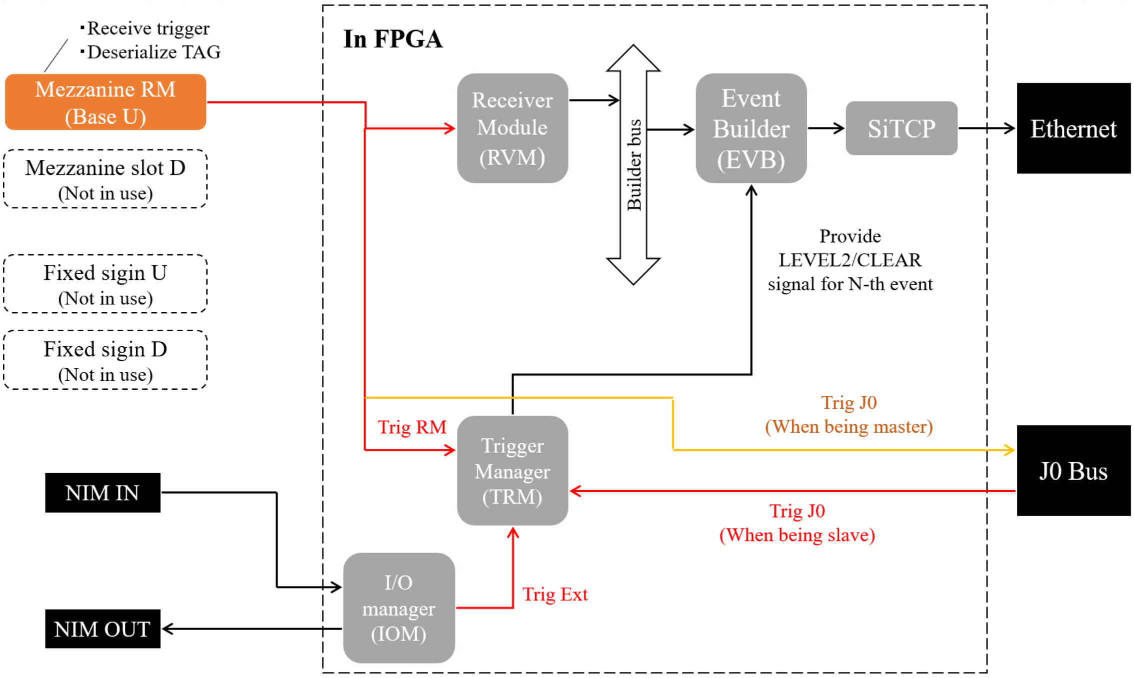

Figure 2: HUL RM firmware block diagram

4.5.1 Register map of HUL RM

The following is a map dedicated to HUL RM. Even if a module or register with the same name exists in other firmware, it does not necessarily have the same address. Be sure to set according to this map (or RegisterMap.hh and namespace of the distributed software).

Register	Address	Operation	bit width	description
		Trigger Manager: TRM (module ID = 0x00)		
SelectTrigger	0x0000'0000	R/W	12	Selects trigger port in TRM
		DAQ Controller: DCT (module ID = 0x01)		
DaqGate	0x1000'0000	R/W	1	ON/OFF of DAQ gate. TRM disables trigger out if zero.
EvbReset	0x1010'0000	W	-	Write to this address asserts a soft reset to EVB, and self event counter in Event builder becomes zero. (Don't care about the register value.)
		IO Manager: IOM (module ID = 0x02)		
NimOut1	0x2000'0000	R/W	4	Determines what to send to NIMOUT1.
NimOut2	0x2010'0000	R/W	4	Determines what to send to NIMOUT2.
NimOut3	0x2020'0000	R/W	4	Determines what to send to NIMOUT3.
NimOut4	0x2030'0000	R/W	4	Determines what to send to NIMOUT4.
ExtL1	0x2040'0000	R/W	3	Determines which NIMIN is connected to ExtL1.
ExtL2	0x2050'0000	R/W	3	Determines which NIMIN is connected to ExtL2.
ExtClr	0x2060'0000	R/W	3	Determines which NIMIN is connected to Ext clear.
ExtBusy	0x2070'0000	R/W	3	Determines which NIMIN is connected to Ext busy.
ExtRsv2	0x2080'0000	R/W	3	Determines which NIMIN is connected to Ext rsv2.

Trigger Manager (TRM)

TRM decides which input port signal to be used as a trigger, and sends L1, L2, and Clear signals to the FPGA. Also, Tag signal is repeated, if received, for a redistribution in FPGA. Which port signal is selected is set by the 12-bit register `SelectTrigger`. The relationship between the trigger signal path and `SelectTrigger` is summarized in Figure. Which port receives the L1 trigger is determined by the 3 bits. Please note that if two or more bits are set, the trigger will come out in OR. Once the L1 route is determined, it will be ANDed with the DAQ gate (DAQ controller management) and distributed as the L1 trigger. The selection of L2 trigger and Clear is also performed with the 3 bits, but these are affected by EnL2 bit after routing. If EnL2 is 0, L2 contains a copy of L1 and Clear is always 0. In a simple system without MTM-RM, set EnL2 to 0. L2 is distributed as L2 trigger after ANDed with the DAQ gate. Tag information source is selected between J0 or HRM using EnJ0 and EnRM bits, respectively. If both are set to 1, Tag will not be issued.

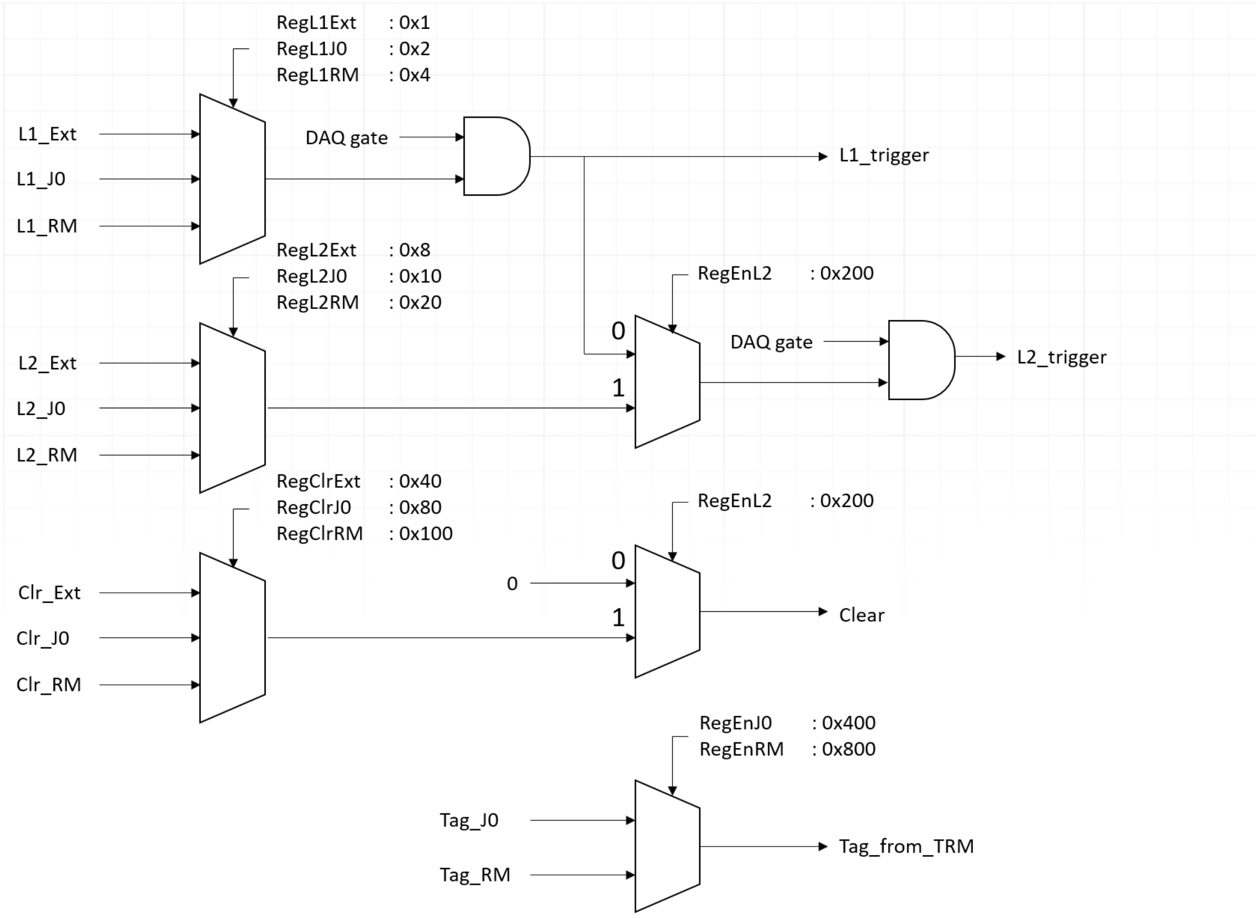

Figure 3: Trigger route in TRM

A list of register values to be stored in `TRM::SelectTrigger` address. Since each bit is a switch for the appropriate selector, this register is a 12-bit wide bit string instead of an integer value. Exceptionally, only `RegEnJ0` is used outside of TRM. Only when `RegEnJ0` is high and DIP SW2 No. 2 (mezzanine HRM) is low, module busy is sent to J0 bus. If you want to insert a module into the crate but do not want to affect the J0 bus, set this register `RegEnJ0` to 0.

Register label	Register value	description
RegL1Ext	0x001	NIMIN provides L1 trigger.
RegL1J0	0x002	J0 bus provides L1 trigger.
RegL1RM	0x004	Mezzanine HRM provides L1 trigger.
RegL2Ext	0x008	NIMIN provides L2 trigger.
RegL2J0	0x010	J0 bus provides L2 trigger.
RegL2RM	0x020	Mezzanine HRM provides L2 trigger.
RegClrExt	0x040	NIMIN provides Clear.
RegClrJ0	0x080	J0 bus provides Clear.
RegClrRM	0x100	Mezzanine HRM provides Clear.
RegEnL2	0x200	0: L2=L1 trigger、1: L2=L2 input
RegEnJ0	0x400	Tag information from J0 bus. If this bit is 1, module busy is sent to J0 bus.
RegEnRM	0x800	Tag information from HRM.

I/O Manager (IOM)

IOM has the function of assigning the signal inside the FPGA to NIMIN or NIMOUT. For example, if Reg_o_ModuleBusy is set to AddrNnimout1, the BUSY signal will be output to NIM output 1 on the front panel. If Reg_i_Nimin1 is set for AddrExtL1, NIM input No. 1 is assigned to the L1Ext line of TRM. The register values that may be stored in the register address are summarized below. The syntax of assignments are opposite between NIMOUTs and NIMINs; NIMOUTs has the address where the signal register value is written in, whereas for inputs, signal has the address where NIMIN values is written in. The register values are interpreted as integers, meaning exclusive with each other.

Register label	Register value	description
		Signals available for NIMOUT
Reg_o_ModuleBusy	0x0	Module busy, meaning only the busy status of the module. J0 bus busy or ExtBusy are not included.
Reg_o_CrateBusy	0x1	CrateBusy, including the module busy, J0 bus busy and ExtBusy. Usage of this signal assumes that HUL is the bus master, and HRM returns the same busy to master trigger module (MTM).
Reg_o_RML1	0x2	HRM L1 trigger as HRM has received.
Reg_o_RML2	0x3	L2 trigger as HRM has received.
Reg_o_RMClr	0x4	Clear as HRM has received.
Reg_o_RMRsv1	0x5	Rserve 1 signal as HRM has received.
Reg_o_RMSnInc	0x6	Spill Number Increment of HRM
Reg_o_DaqGate	0x7	DAQ gate in DCT
Reg_o_DIP8	0x8	ch 8 of DIP SW2
Reg_o_clk1MHz	0x9	1 MHz clock
Reg_o_clk100kHz	0xA	100 kHz clock
Reg_o_clk10kHz	0xB	10 kHz clock
Reg_o_clk1kHz	0xC	1 kHz clock
		NIMIN ports available
Reg_i_nimin1	0x0	NIMIN1
Reg_i_nimin2	0x1	NIMIN2
Reg_i_nimin3	0x2	NIMIN3
Reg_i_nimin4	0x3	NIMIN4
Reg_i_default	0x7	If this register value is set, the default assignment are done for signal lines (see next table), including the NIMOUTs.

IOM default assignments as listed below.

NIM output ports	Register	
NIMOUT1	Reg_o_ModuleBusy	
NIMOUT2	reg_o_DaqGate	
NIMOUT3	reg_o_clk1kHz	
NIMOUT4	reg_o_DIP8	
Signal	**Register**	**default**
ExtL1	Reg_i_Nimin1	NIMIN1
ExtL2	Reg_i_default	0
ExtLClear	Reg_i_default	0
ExtLBusy	Reg_i_nimin3	NIMIN3
ExtLRsv2	Reg_i_nimin4	NIMIN4

4.5.2 DIP SW and LED on HUL RM

DIP SW2 functions

Switch number	function	detail
1	SiTCP force default	ON for SiTCP forced default (192.168.10.16). Must be set before power on.
2	Mezzanine HRM	ON for Mezzanine HRM installed. The actual effect by this mode is described later. This bit status appears in data header3.
3	Force BUSY	Forced high for Crate Busy and Module Busy. Used for connection check.
4	Bus BUSY	ON to include J0 bus busy to Crate Busy. OFF otherwise.
5	LED	ON to turn on LED4.
6	Not in Use	
7	Not in Use	
8	Level	Appears as IOM DIP8 signal.

The effect of Mezzanine HRM (DIP SW2 #2)

If this bit is ON, a few functions change with regard to Mezzanine HRM.

Event Builder includes RVM in the event packet.

If this bit is ON, the information in RVM is read by Event Builder and includes into the event packet.

Mezzanine base (U) signal direction change

If this bit is ON, a few signal lines to be LVDS output as required by Mezzanine HRM.
if this bit is OFF, all the slot lines will become LVDS inputs.

J0 bus master mode to be ON

If this bit is ON, L1, L2, Clear, Tag information are sent to the J0 bus, and the BUSY signal is received from J0 bus. To be a J0 bus master, it is also necessary to turn all DIP SW1 ON.

J0 bus slave mode to be OFF
If this bit is OFF, L1, L2, Clear, Tag will not be accepted from J0 bus.

Below is the relation between J0 bus signals and the trigger signals.

J0 bus	Trigger signal
S1	RM_Clear
S2	RM_Level2
S3	RM_SpillNumber(0)
S4	RM_Level1
S5	RM_EventNumber(0)
S6	RM_EventNumber(1)
S7	RM_EventNumber(2)

LED indication

LED number	description
LED1	Light when TCP connection is open.
LED2	Light when module busy is high.
LED3	Light when DIP SW2 #2 (Mezzanine HRM) is ON.
LED4	Light when DIP SW2 #5 (LED) is ON.

4.5.3 DAQ behavior of HUL RM

This section describes data flow and DAQ behavior. The DAQ function consists of each measurement module (hereinafter referred to as the measurement block) and the Event Builder module. The data flow is shown in Figure. When the trigger is received, each measurement block processes the data according to the determined operation and saves it in the block buffer. The measurement block has an internal block buffer that can temporarily store multi-events. Event Builder reads the data from each measurement block and continues to build events unless the event buffer is full. Therefore, the DAQ function operates synchronously with the trigger until the data is written to the block buffer; the subsequent processing does not depend on the external signals nor states and continues to build and transfer the data as long as the data link speed allows. For HUL RM, the measurement blocks are only RVM and TRM. Strictly speaking, TRM is not a measurement block because the TRM information is not included in the data body but in the header. The only event-built information is RVM data, and the TRM information is used to control data transfer. The TRM stores whether the L2 trigger or Clear was sent in the Nth event, and this information is used by the Event Builder to decide whether to send this event packet to SiTCP or just drop it. Therefore, HUL's DAQ function does not have a fast clear nor clear BUSY functions. All events raised by the L1 trigger are digitized and built once. However, the self-event number assigned by Event Builder is not incremented unless it is forwarded.

RVM latches the information shown in Figure at the timing of L2 trigger and saves it in the block buffer.

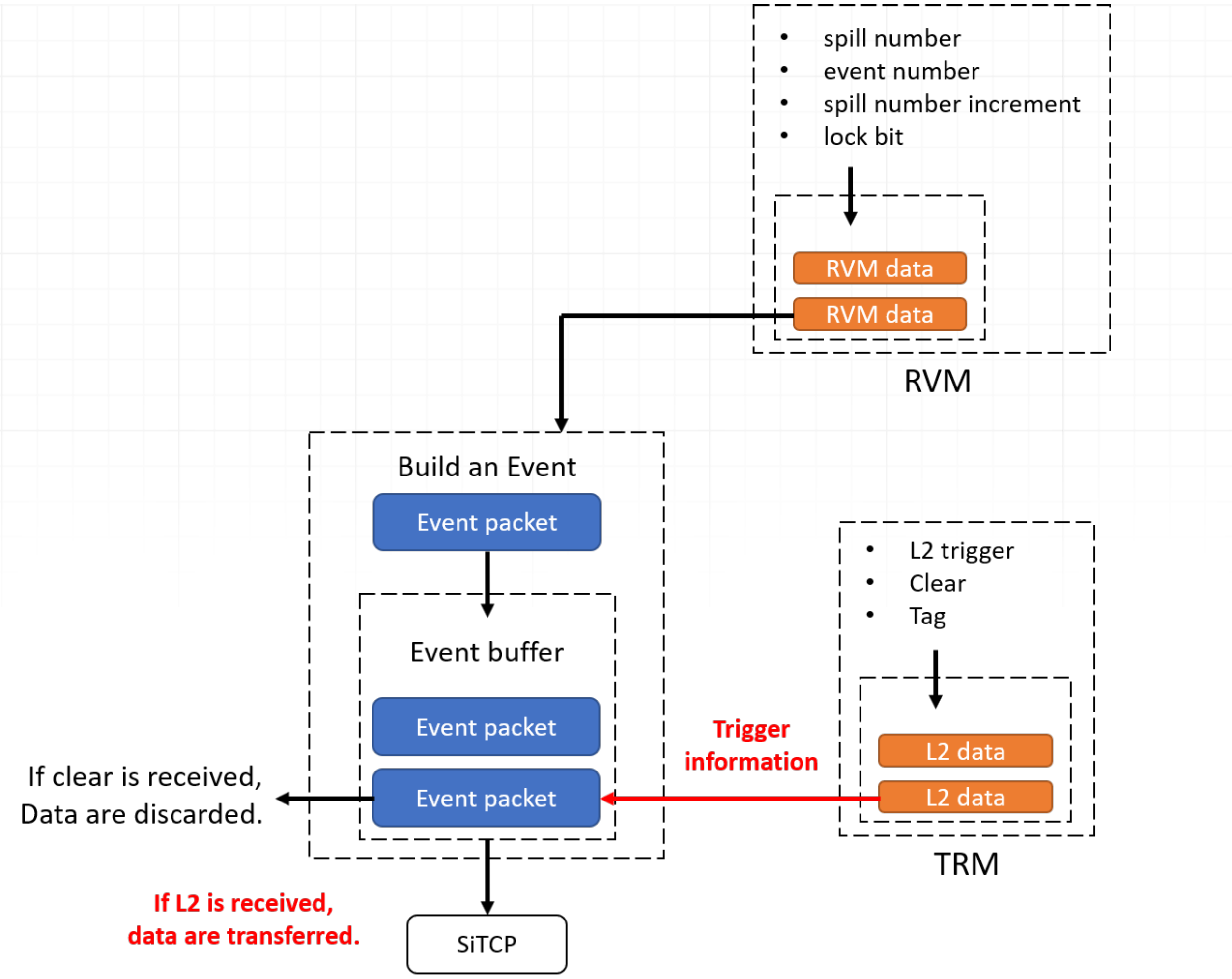

Figure 4: DAQ data flow of HUL RM firmware

The timing when Module Busy is asserted

The definition of Module BUSY in HUL RM is the OR of the BUSY signals listed below. Block full and SiTCP full occurs only when network forwarding can not keep up, so usually BUSY is a fixed length of 160 ns. Currently, Self-busy, set to 160 ns, is rather long, and it may be shortened in the future.

BUSY type	BUSY length	memo
Self busy	160 ns	Asserted with a fixed length since the L1 trigger is detected.
Block full	-	BUSY is output when the block buffer is full. It is asserted when the L1 trigger rate exceeds the data processing speed of the subsequent circuit. In other words, it means that TCP transfer cannot catch up, so it is practically equivalent to SiTCP full.
SiTCP full	-	Asserted when the TCP buffer of SiTCP is Full, meaning that the amount of data which the Event Builder is trying to send is too large for the network bandwidth.

Data structure

In HUL, 32-bit is one word, and the three-word header and variable-length data body are one event block.

Header word

```
Header1 (Magic word)
MSB                                                                                      LSB
|                                          0xFFFF0415                                      |

Header2 (event size)
|       0xFF       |       0x00       |     "00000"    |          Number of word (11-bit)          |
```

"Number of word" indicates the number of words contained in the data body, not including the header.

```
Header3 (event number)
|       0xFF       | HRM exist | "000" | Tag (4-bit) |          Self counter (16-bit)           |
```

If *HRM* exist is 1, ch2 of DIP SW2 is on, meaning HRM was installed. This means that data body has a RVM word. *Tag* " is the 4-bit Tag information from TRM. The lower 3 bits are the lower 3 bits of the RM Event Number, and the 4th bit is the least significant bit of the RM spill number. "Self counter" is a local event number that is incremented each time an event is forwarded. Starts with 0.

Data body

```
RVM word
|  0xF9   | "00"  | Lock  | SNI  | Spill Num (8-bit) |            Event Num (12-bit)            |
```

Lock means the RM lock bit, must be 1. *SNI* means Spill Number Increment, which becomes 1 at the change of the Spill Number (not tested if it is true). *Spill Num* is the spill number, and *Event Num* is the event number, received by HRM, respectively.

4.6 HUL Scaler

HUL Scaler is a firmware that adds a scaler function to HUL RM. The implemented scaler is a 28-bit synchronization counter that samples at 300 MHz. Since the HUL Scaler has many functions in common with the HUL RM, only the differences will be mentioned. **Firmware ID and current version**

ID	0x4ca1
Major version	0x03
Minor version	0x03

Version history

Exactly same with HRM, except the current version is v3.3.

Overview of Module functions

HUL Scaler is implemented for Mezzanine slot D and on-board input ports, in addition to Mezzanine slot U configured for HRM. DCR v1 (v2) is assumed to be installed in the Mezzanine slot(s), implementing scalers up to 128 channels. It is also possible to mount HRM (instead of DCR) on Mezzanine slot U to become a J0 bus master like HUL RM firmware. In this case, the Ch 0-31 assigned to slot U is deleted from the data.

The Scaler consists of a 300 MHz, 28-bit long counter that latches the counter at the timing of the L1 trigger and writes it to the buffer. The HUL scaler has two new internal signals connected to the IOM: one is the *spill gate*, which enables the scalers only while this signal is high. The other is *counter reset*, which resets all counts to zero when it becomes high. Enable/Disable NIM input counter reset is set with *enable_hdrst* (in v1.6 and later). Other features are common to HUL RM.

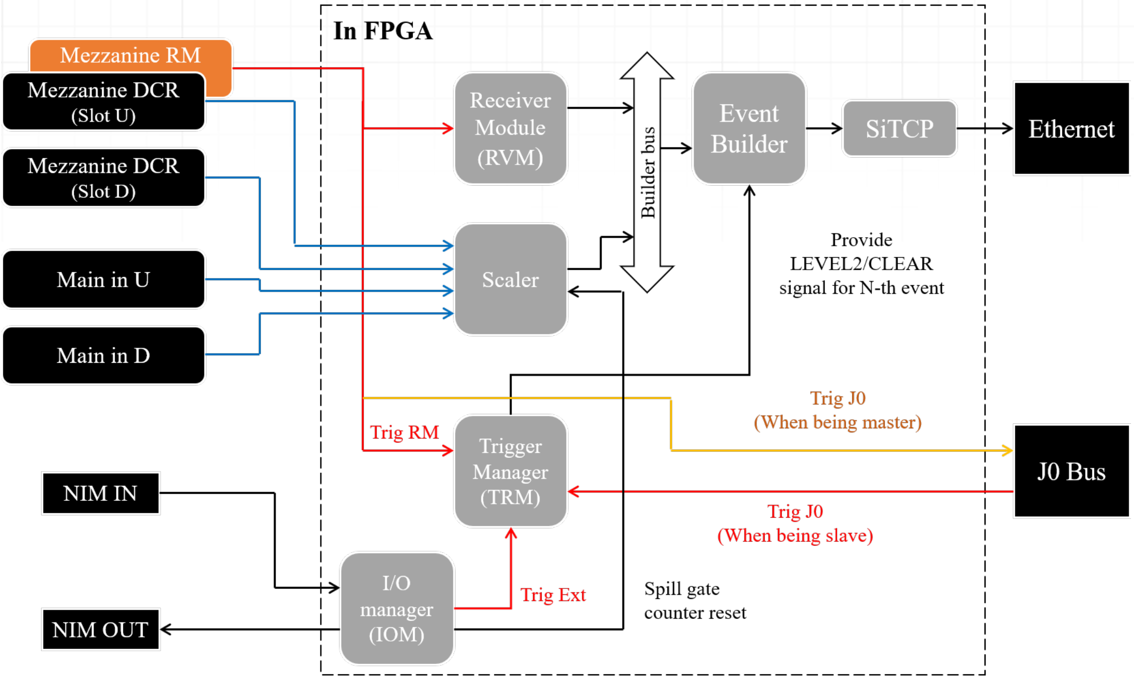

Figure 5: Structure of HUL Scaler firmware

4.6.1 Register map for HUL Scaler

The following is a map dedicated to HUL Scaler. The difference in the signal and address from HUL RM is marked as red . Signal names and the address are defined in RegisterMap.hh and namespace in the software package.

Register	Address	Operation	bit width	Description
		Trigger Manager: TRM (module ID = 0x00)		
SelectTrigger	0x0000'0000	R/W	12	Selects trigger port in TRM
		DAQ Controller: DCT (module ID = 0x01)		
DaqGate	0x1000'0000	R/W	1	ON/OFF of DAQ gate. TRM disables trigger out if zero.
EvbReset	0x1010'0000	W	-	Write to this address asserts a soft reset to EVB, and self event counter in Event builder becomes zero. (Don't care about the register value.)
		IO Manager: SCR (module ID = 0x02)		
CounterReset	0x2000'0000	W	-	Asserts Software counter reset
EnableBlock	0x2010'0000	R/W	4	Enable/Disable the input blocks. High for enable. The bit corresponds to: bit1: On-board U bit2: On-board D bit3: Mezzanine U bit4: Mezzanine D
EnableHdrst	0x2020'0000	R/W	4	Enable/Disable the NIM input hardware counter reset for input blocks. High for enable. The bit corresponds to: bit1: On-board U bit2: On-board D bit3: Mezzanine U bit4: Mezzanine D
		IO Manager: IOM (module ID = 0x03)		
NimOut1	0x3000'0000	R/W	4	Determines what to send to NIMOUT1.
NimOut2	0x3010'0000	R/W	4	Determines what to send to NIMOUT2.
NimOut3	0x3020'0000	R/W	4	Determines what to send to NIMOUT3.
NimOut4	0x3030'0000	R/W	4	Determines what to send to NIMOUT4.
ExtL1	0x3040'0000	R/W	3	Determines which NIMIN is connected to ExtL1.
ExtL2	0x3050'0000	R/W	3	Determines which NIMIN is connected to ExtL2.
ExtClr	0x3060'0000	R/W	3	Determines which NIMIN is connected to Ext clear.
ExtSpillGate	0x3070'0000	R/W	3	Determines which NIMIN is connected to ext spill gate.
ExtCCRst	0x3080'0000	R/W	3	Determines which NIMIN is connected to ext counter reset
ExtBusy	0x3090'0000	R/W	3	Determines which NIMIN is connected to Ext busy.
ExtRsv2	0x30A0'0000	R/W	3	Determines which NIMIN is connected to Ext rsv2.

4.6.2 Function of each block on HUL Scaler

Trigger Manager (TRM)

Same with HUL RM.

Scaler (SCR)

The scaler is the main function of the HUL Scaler. Each scaler unit synchronizes the input signal at 300 MHz, then performs edge detection and increments the counter by one at that edge timing. The minimum width of the input pulse is 3.5 ~ 4.0 ns, and the minimum separation between two pulses is about the same. The counter is 28-bit long and returns to 0 after going around. The scaler is divided into 4 blocks of 32ch, and they are enable/disabled in the `EnableBlock` register. If the corresponding bit is high/low, the entire block (32ch) is enabled/disabled. The scaler is reset to zero by the hard reset `ExtCounterReset` (controlled by NIMIN, IOM) or the soft reset `CounterReset` is asserted. The hard reset is enabled/disabled by the block with `EnableHdrst`. If this bit is 1, the counter will be 0 at the timing of the hard reset. The operation is undefined when a reset is entered within 100 ns of the trigger. The data may be returned, but contain unintended values.

The entire scalers are enabled/disabled by `ExtSpillGate` (NIMIN, controlled by IOM). The scaler is only incremented when the spill gate is high. The Spill gate is 1 by default, and may be assigned to an NIM input port. **I/O Manager (IOM)**

IOM is the same with HUL RM, with additional NIM inputs `ExtSpillGate` and `ExtCCRst`. No change NIMOUTs.

Register label	Register value	description
		Signals available for NIMOUT
Reg_o_ModuleBusy	0x0	Module busy, meaning only the busy status of the module. J0 bus busy or ExtBusy are not included.
Reg_o_CrateBusy	0x1	CrateBusy, including the module busy, J0 bus busy and ExtBusy. Usage of this signal assumes that HUL is the bus master, and HRM returns the same busy to master trigger module (MTM).
Reg_o_RML1	0x2	HRM L1 trigger as HRM has received.
Reg_o_RML2	0x3	HRM L2 trigger as HRM has received.
Reg_o_RMClr	0x4	HRM Clear as HRM has received.
Reg_o_RMRsv1	0x5	HRM Reserve 1 as HRM has received.
Reg_o_RMSnInc	0x6	HRM outputs Spill Number Increment
Reg_o_DaqGate	0x7	DAQ gate of DCT
Reg_o_DIP8	0x8	ch 8 of DIP SW2
Reg_o_clk1MHz	0x9	1 MHz clock
Reg_o_clk100kHz	0xA	100 kHz clock
Reg_o_clk10kHz	0xB	10 kHz clock
Reg_o_clk1kHz	0xC	1 kHz clock
		NIMIN ports available
Reg_i_nimin1	0x0	NIMIN1
Reg_i_nimin2	0x1	NIMIN2
Reg_i_nimin3	0x2	NIMIN3
Reg_i_nimin4	0x3	NIMIN4
Reg_i_default	0x7	If this register is set, the default assignment are done for signal lines (see next table).

IOM default assignments as listed below.

"

NIM output ports	Register
NIMOUT1	Reg_o_ModuleBusy
NIMOUT2	reg_o_DaqGate
NIMOUT3	reg_o_clk1kHz
NIMOUT4	reg_o_DIP8

Signal	Register	default
ExtL1	Reg_i_Nimin1	NIMIN1
ExtL2	Reg_i_default	0
ExtLClear	Reg_i_default	0
ExtSpillGate	Reg_i_default	1
ExtCounterReset	Reg_i_nimin2	NIMIN2
ExtLBusy	Reg_i_nimin3	NIMIN3
ExtLRsv2	Reg_i_nimin4	NIMIN4

4.6.3 Switch and LED on HUL Scaler

DIP SW2

The same with HUL RM.

LED

The same with HUL RM.

4.6.4 DAQ behavior of HUL Scaler

The data flow is shown in Figure. If HRM is installed and DIP SW2 ch2 (mezzanine HRM) is ON, RVM and SCR contributes to the event; otherwise, only SCR contributes. The *number of word* of header2 contains the total of SCR and RVM data.

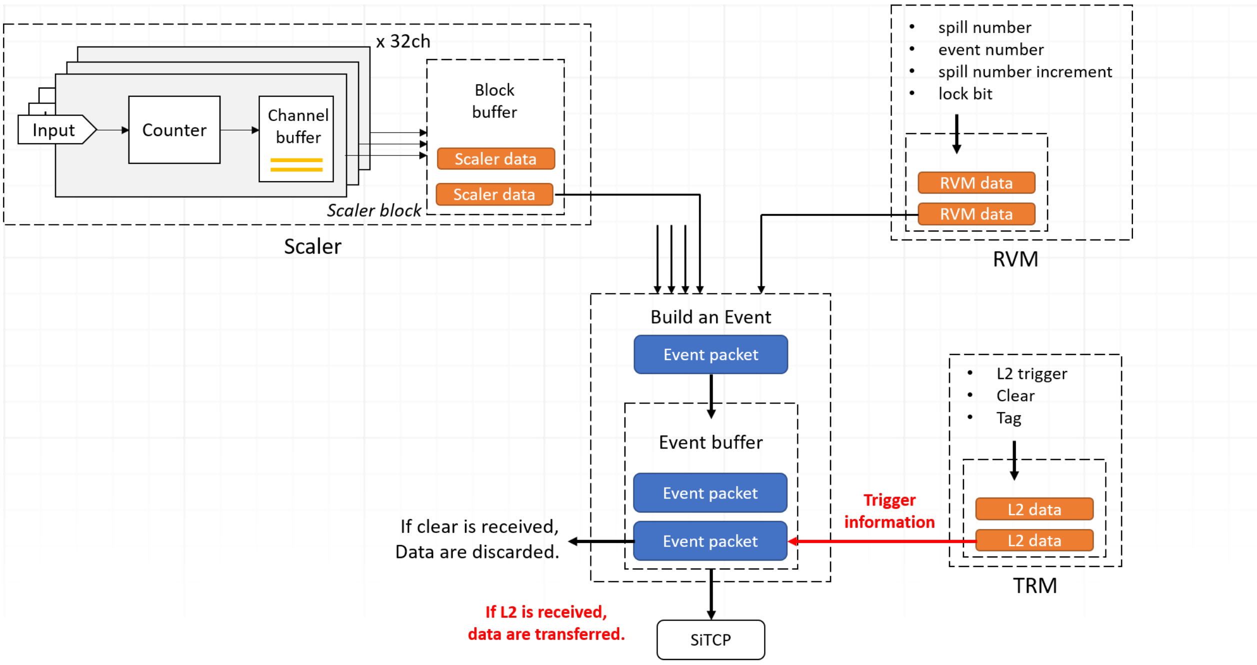

Figure 6: DAQ data path of HUL Scaler firmware

Module Busy timing

HUL Scaler has the same definition of module BUSY with HUL RM, except the time length is different (HUL Scaler: 210 ns, HUL RM: 160 ns) due to the difference of the system clock frequency.

Data structure

Header word

```
Header1 (Magic word)
MSB                                                                                    LSB
|                                       0xFFFF4CA1                                     |

Header2 (event size)
|       0xFF       |       0x00       |     "00000"    |         Number of word (11-bit)          |
```

"Number of word" indicates the number of words contained in the data body, not including the header.

```
Header3 (event number)
|       0xFF       | HRM exist | "000" | Tag (4-bit) |         Self counter (16-bit)          |
```

If *HRM* exist is 1, ch2 of DIP SW2 is on, meaning HRM was installed. This means that data body has a RVM word. *Tag* is the 4-bit Tag information from TRM. The lower 3 bits are the lower 3 bits of the RM Event Number, and the 4th bit is the least significant bit of the RM spill number. *Self counter* is a local event number that is incremented each time an event is forwarded. Starts with 0.

Data body

```
RVM word
|  0xF9    | "00"   | Lock   | SNI   | Spill Num (8-bit) |            Event Num (12-bit)         |
```

Lock means the RM lock bit, must be 1. *SNI* means Spill Number Increment, which becomes 1 at the change of the Spill Number (not tested if it is true). *Spill Num* is the spill number, and *Event Num* is the event number, received by HRM, respectively.

```
SCR word
|  SCR block (4-bit)   |                              Counter (28-bit)                              |
```

SCR Block indicates which input block the word belongs to. There is no field to indicate the channel in the SCR words; they are aligned from lower to higher channel numbers in the input block.

SCR block bits	Input block
0x8	Main input port U
0x9	Main input port D
0xA	Mezzanine U
0xB	Mezzanine D

4.7 HUL MH-TDC

HUL MH-TDC is a firmware that adds a multi-hit TDC function to HUL RM. Since the HUL MH-TDC has many functions in common with the HUL RM, only the differences will be mentioned.

Firmware ID and current version

ID	0x30CC
Major version	0x03
Minor version	0x04

Version history Similar to HUL RM version history. Marked red if different.

Version	Release date	Modifications
v1.0	2016.12.23	Initial release
v1.1	2017.01.15	RVM data header changed from 0x9C to 0xF9.
v1.2	2017.01.27	Vivado更新 2016.2 => 2016.4。Block buffer changed from BuildIn FIFO to BRAM with depth 4096. EventBuffer depth changed from 2048 to 4096, and pgfull to be 4058. TDC block channel buffer changed from dispersive RAM to BRAM. TRM middle buffer changed from disperse RAM to BRAM. The depth changed from 128 to 256. Prog Full threshold introduced. RVM middle buffer depth is also changed to 256 (128?).
v1.4	2017.05.09	Solved the problem of not responding to Clear (BUSY stays standing).
v1.5	-	Solved the problem that HRM hangs when Clear is entered. (Replaced by v1.6 without release.)
v1.6	-	Addressed the issue that the event ID shifts when max multihit (16 hit / ch) is reached once. (un-released)
v1.7	2017.08.22	Bug fix of event sequence in FPGA
v1.8	2017.12.19	Standardlized reset sequence. Bit 24 of Header3 is now indicating whether HRM exists (to be exact, whether DIP2 is ON). Bug fix of rare broken data in high count-rate. Number of words in Header2 changed the width from 11-bit to 12-bit.
v1.9	2018.02.02	>Solved the bug that the event tag coming from the J0 bus was latched too early and the event number on the HRM side deviated by 1. Solved the bug that data comes back out side of the search window (common to Mezzanine HR-TDC).
v2.x	-	un-released
v3.4	2021.08.01	Added FMP and SDS. Installed Builder bus. Changed the structure of Local bus.

Overview of Module functions

HUL MH-TDC is implemented for Mezzanine slot D and on-board input ports, in addition to Mezzanine slot U configured for HRM. DCR v1 (v2) is assumed to be installed in the Mezzanine slot(s), implementing TDC up to 128 channels. It is also possible to mount HRM (instead of DCR) on Mezzanine slot U to become a J0 bus master like HUL RM firmware. In this case, the Ch 0-31 assigned to slot U is deleted from the data.

The MH-TDC implements a 300 MHz 4-phase clock TDC with a 1-bit precision of 0.83 ns. Both leading and trailing edges can be detected, the length of time that can be traced back from the trigger is 13.7 us, and the timing resolution is 300 ps (σ).

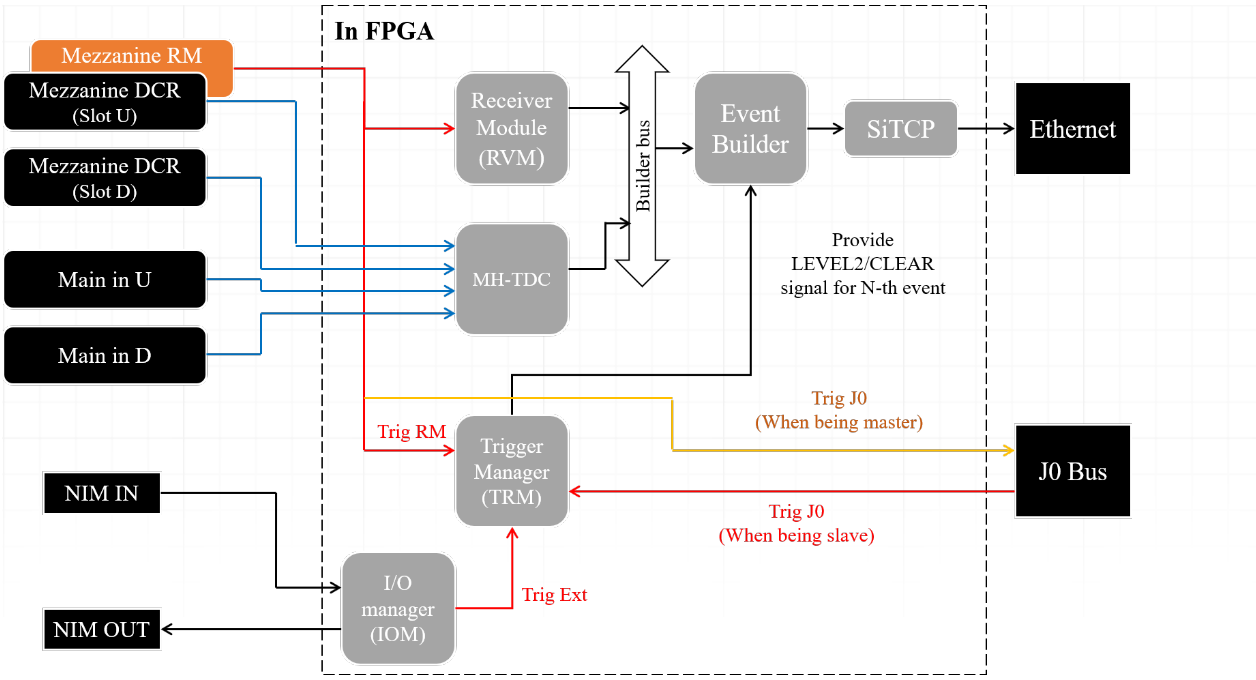

Figure 7: HUL MH-TDC FWの構造。

4.7.1 Register map of HUL MH-TDC

The following is a map dedicated to HUL MH-TDC. The difference in the signal and address from HUL RM is marked as red . Signal names and the address are defined in RegisterMap.hh and namespace in the software package.

Register	Address	Operation	bit width	Description
Trigger Manager: TRM (module ID = 0x00)				
SelectTrigger	0x0000'0000	R/W	12	Selects trigger port in TRM
DAQ Controller: DCT (module ID = 0x01)				
DaqGate	0x1000'0000	R/W	1	On/Off od the DAQ gate.
EvbReset	0x1010'0000	W	-	Write to this address asserts a soft reset to EVB, and self event counter in Event builder becomes zero. (Don't care about the register value.)
IO Manager: TDC (module ID = 0x02)				
EnableBlock	0x2000'0000	R/W	4	Enable/Disable the input blocks. High for enable. The bit corresponds to: 1st bit : Main input port U 2nd bit : Main input port D 3rd bit : Mezzanine U 4th bit : Mezzanine D
PtrOfs	0x2010'0000	R/W	11	Internal use. Do not touch.
WindowMax	0x2020'0000	R/W	11	The upper limit of the time window to search for hits from the Ring buffer. 1bit is equivalent to 6.666... ns. Detail is described later.
WindowMin	0x2030'0000	R/W	11	The lower limit of the time window to search for hits from the Ring buffer. 1bit is equivalent to 6.666... ns. Detail is described later.
IO Manager: IOM (module ID = 0x03)				
NimOut1	0x3000'0000	R/W	4	Determines what to send to NIMOUT1.
NimOut2	0x3010'0000	R/W	4	Determines what to send to NIMOUT2.
NimOut3	0x3020'0000	R/W	4	Determines what to send to NIMOUT3.
NimOut4	0x3030'0000	R/W	4	Determines what to send to NIMOUT4.
ExtL1	0x3040'0000	R/W	3	Determines which NIMIN is connected to ExtL1.
ExtL2	0x3050'0000	R/W	3	Determines which NIMIN is connected to ExtL2.
ExtClr	0x3060'0000	R/W	3	Determines which NIMIN is connected to Ext clear.
ExtBusy	0x3070'0000	R/W	3	Determines which NIMIN is connected to Ext busy.
ExtRsv2	0x3080'0000	R/W	3	Determines which NIMIN is connected to Ext rsv2.

4.7.2 Function of each block on HUL MH-TDC

Trigger Manager (TRM)

Same with HUL RM.

Multi-Hit TDC (MH-TDC)

This is the main function of this firmware. This MH-TDC uses a 4-phase clock to create a pseudo 1.2 GHz. The multi-hit TDC block shown in Figure contains three components: the TDC unit, the ring buffer, and the channel buffer. The TDC unit measures time at pseudo 1.2 GHz and performs hit detection. The TDC unit has a time resolution of 300 ps (σ) and a minimum detectable pulse width of approximately 4 ns. The detected hit information is saved in the ring buffer. The length of the ring buffer is 13.7 us, and the write / read pointer of the ring buffer corresponds to the course count. The Ring buffer is driven by a 150 MHz clock, so the course count is 6.666 ... ns accurate.

When a L1 trigger is detected, ring buffer read out starts. The range to search for hits is set by `WindowMax` and `WindowMin` registers. These registers are 11-bit integer with the course count resolution for 1-bit. The Hits which do not fall within the range will not be written to the channel buffer. BUSY is asserted while searching for hit information from the Ring buffer.

The maximum number of hits allowed for 1ch / event is 16 hits, when transferred from channel buffer to block buffer in reverse time order. Any hits more than that will be discarded and the overflow bit will be set.

Multi-hit TDC specfication	
TDC resolution	0.833... ns
coarse count resolution	6.66... ns
Ring buffer length	13.8 us
timing resolution	300 ps (σ) *measured
minimum pulse width	~4 ns
double hit resolution	~7 ns
maximum hits/ch/event	16

I/O Manager (IOM) IOM is the same with HUL RM.

Register label	Register value	description
		Signals available for NIMOUT
Reg_o_ModuleBusy	0x0	Module busy, meaning only the busy status of the module. J0 bus busy or ExtBusy are not included.
Reg_o_CrateBusy	0x1	CrateBusy, including the module busy, J0 bus busy and ExtBusy. Usage of this signal assumes that HUL is the bus master, and HRM returns the same busy to master trigger module (MTM).
Reg_o_RML1	0x2	HRM L1 trigger as HRM has received.
Reg_o_RML2	0x3	HRM L2 trigger as HRM has received.
Reg_o_RMClr	0x4	HRM Clear as HRM has received.
Reg_o_RMRsv1	0x5	HRM Reserve 1 as HRM has received.
Reg_o_RMSnInc	0x6	HRM Spill Number Increment as HRM has received
Reg_o_DaqGate	0x7	DAQ gate in DCT
Reg_o_DIP8	0x8	ch 8 of DIP SW2
Reg_o_clk1MHz	0x9	1 MHz clock
Reg_o_clk100kHz	0xA	100 kHz clock
Reg_o_clk10kHz	0xB	10 kHz clock
Reg_o_clk1kHz	0xC	1 kHz clock
		NIMIN ports available
Reg_i_nimin1	0x0	NIMIN1
Reg_i_nimin2	0x1	NIMIN2
Reg_i_nimin3	0x2	NIMIN3
Reg_i_nimin4	0x3	NIMIN4
Reg_i_default	0x7	If this register is set, the default assignment are done for signal lines (see next table).

IOM default assignments as listed below.

NIM output ports	Register	
NIMOUT1	Reg_o_ModuleBusy	
NIMOUT2	reg_o_DaqGate	
NIMOUT3	reg_o_clk1kHz	
NIMOUT4	reg_o_DIP8	
Signal	**Register**	**default**
ExtL1	Reg_i_Nimin1	NIMIN1
ExtL2	Reg_i_default	0
ExtLClear	Reg_i_default	0
ExtLBusy	Reg_i_nimin3	NIMIN3
ExtLRsv2	Reg_i_nimin4	NIMIN4

4.7.3 Switch and LED on HUL MH-TDC

DIP SW2

The same with HUL RM

LED

The same with HUL RM

4.7.4 DAQ operation

Data flow is shown in Figure. If HRM is installed and DIP SW2 ch2 (mezzanine HRM) is ON, RVM and TDC contributes to the event; otherwise, only TDC contributes. The "number of word" of header2 contains the total of TDC and RVM data.

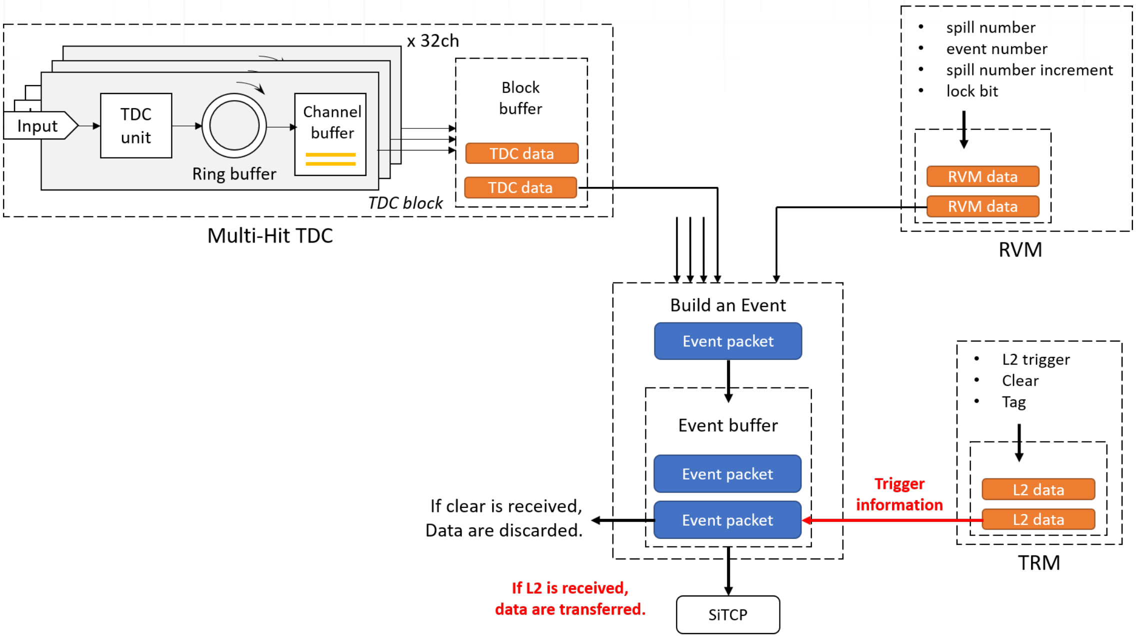

Figure 8: DAQ data flow of HUL MH-TDC firmware

Module Busy timing

HUL MH-TDC has the same definition of module BUSY with HUL RM, except the time length is different (HUL MH-TDC: 210 ns, HUL RM: 160 ns) due to the difference of the system clock frequency. In addition it as the Sequence busy, which is ORed.

BUSY type	BUSY length	description
Self busy	210 ns	Asserted with a fixed length since the L1 trigger is detected.
Sequence busy	Dependes on the search window length.	Depends on \`WindowMax\`-\`WindowMin\`: BUSY while hits are searched in the Ring buffer.
Block full	-	BUSY is output when the block buffer is full. It is asserted when the L1 trigger rate exceeds the data processing speed of the subsequent circuit. This happens when TCP transfer cannot catch up and is practically equivalent to SiTCP full.
SiTCP full	-	TCP buffer of SiTCP becomes Full. It is asserted when the amount of data that the Event Builder is trying to send is large for the network bandwidth.

Data structure

Header word

```
Header1 (Magic word)
MSB                                                                                LSB
|                                    0xFFFF30CC                                     |

Header2 (event size)
|      0xFF      |      0x00      |    "0000"    |         Number of word (12-bit)          |
```

Number of word indicates the number of words contained in the data body, not including the header.

```
Header3 (event number)
|      0xFF      | HRM exist | "000" | Tag (4-bit) |        Self counter (16-bit)           |
```

If *HRM* exist is 1, ch2 of DIP SW2 is on, meaning HRM was installed. This means that data body has a RVM word. *Tag* is the 4-bit Tag information from TRM. The lower 3 bits are the lower 3 bits of the RM Event Number, and the 4th bit is the least significant bit of the RM spill number. *Self counter* is a local event number that is incremented each time an event is forwarded. Starts with 0.

Data body

```
RVM word
|  0xF9   | "00"  | Lock  | SNI  | Spill Num (8-bit) |           Event Num (12-bit)           |
```

Lock means the RM lock bit, must be 1. *SNI* means Spill Number Increment, which becomes 1 at the change of the Spill Number (not tested if it is true). *Spill Num* is the spill number, and *Event Num* is the event number, received by HRM, respectively.

```
TDC word
|  Magic word (8-bit)  |  "0" + Ch (7-bit)  | "00" |           TDC (14-bit)                |
```

The "Magic word" is defined as follows.

- 0xCC Leading
- 0xCD Trailing

Ch is the channel number starts with 0, up to 127. Refer to Chapter 1 for Channel-Input port assignment. *TDC* is the 14-bits in the least.

4.8 Mezzanine HR-TDC and HUL HR-TDC BASE

This section describes the firmware inside the Mezzanine HR-TDC and the firmware to control it. Mezzanine HR-TDC is a firmware that implements the functions up to the block buffer in HUL MH-TDC, and HUL HR-TDC BASE implements the subsequent function to manage the entire DAQ such as event builder and trigger manager. Therefore, the Mezzanine HR-TDC cannot perform complicated operations. It transfers the measurement data to the HUL in response to the Trigger (Common stop). The control system is rather complicated because it has two FPGAs. Mezzanine HR-TDC has features that other firmware does not have, such as a tapped-delay-line calibration LUT and DDR communication for data transfer. This section describes these functions and control methods.

Mezzanine HR-TDC ID and current version

Previous firmware has two versions with/without the trailing edge measurements; the new firmware is integrated into one. The edge to be measured is selected by the register.

ID number	0x80cc
Major version	0x05
Minor version	0x00

Version history

Version	Release date	Changes
v2.5	2017.12.19	initial release with leading edge measurements.
v2.6	2018.02.02	Resolved an issue of returning data out of the search window.
v3.2	2017.12.19	Initial release for leading+trailing edge measurements
v3.3	2018.02.02	Resolved an issue of returning data out of the search window.
v4.5	2021.08.01	Installation of SEM and XADC. Installation of Builder bus. Register selection of the measurement edges. Minor bug correction on XDC (the same functions with v4.3). TDC samplning clock and system clock signals are 520 MHz ans 130 MHz up to this version.
v5.0	2023.01.17	Implemented trigger signal output from mezzanine card. Modified the local bus bridge. Mezzanine FW v5.0 is not compatible with HUL HRTDC BASE firmware prior to v3.7. TDC samplning clock and system clock signals are changed to 500 MHz ans 125 MHz, respectively.

Overview of module function

Block diagram of Mezzanine HR-TDC and HUL HR-TDC BASE are shown in Figure, which is drawn in some detail, since the control system is more complicated than other firmware. In the HR-TDC system, BCT exists in each FPGA, and the mezzanine HR-TDC side controls registers with two-step access through the BsuBrige on the BASE side. There is a dedicated C ++ function for controlling the Mezzanine side through BisBridge, which will be explained in more detail in the software section.

The system is divided into two subsystems: Mezzanine HR-TDC which only measures time, and BASE which manages event build, trigger control and IOs. Trigger information is managed by the Trigger Manager (TRM) like any other modules. Only the level 1 trigger that is issued by TRM is sent to the Mezzanine HR-TDC. This signal works as a common stop for the mezzanine HR-TDC. The operation as TDC is equivalent to HUL MH-TDC. Only the time resolution is higher. The ring buffer length for recording hits is 15.7 us, and the time resolution is 25 ps (σ) (for common stop) and 20 ps (σ) (difference between channels).

In order to transfer data from mezzanine to BASE at a high speed, 5 signal lines are used. Since the transfer includes the control bits, not the full bandwidth is available; the time required for transfer of one word (32 bit) is 8 ns (i.e ~4 Gbps). The DDR receiver on the BASE side needs to be initialized once after the power is turned on. The initialization method is also described in the software section. The TDC base receives the data and prepares to pass it to the event builder.

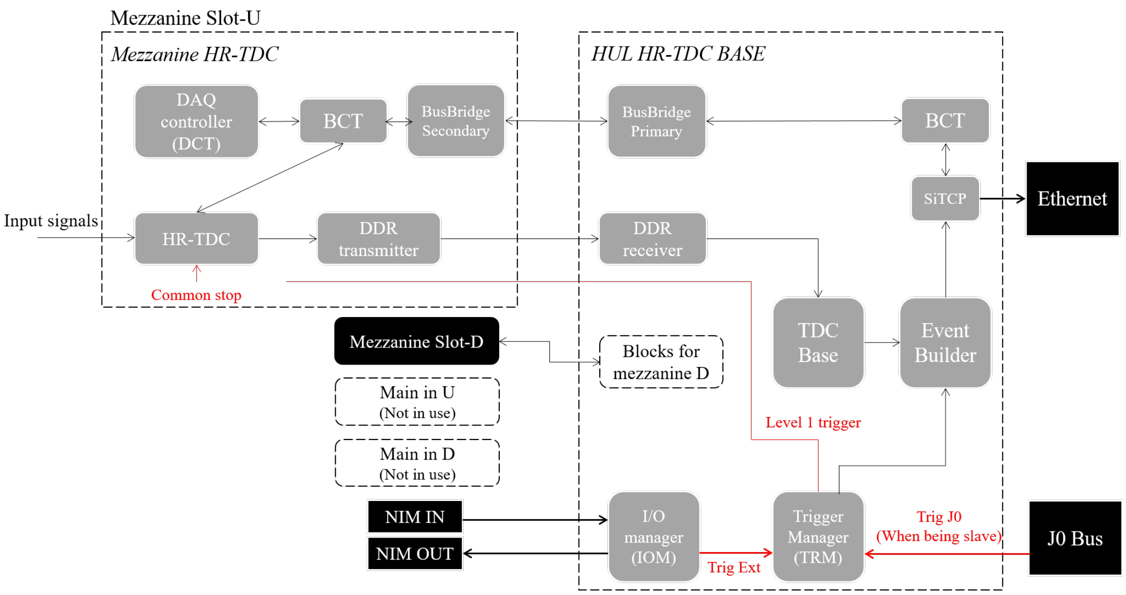

Figure 9: Block diagram of HUL HR-TDC formware

4.8.1 Details of Mezzanine HR-TDC

The principle of high-timing resolution measurements

This firmware uses a time measurement method called the tapped-delay-line (TDL) method. The concept of TDL is shown in Figure. TDL is consist of linear connections of fine delay elements (dTs) and flip-flops (FFs); the time information is interpolated by observing how many FFs has the input signal edge run through. The gray square in Figure shows the delay elements, and the D-FF array takes a snapshot of the FF on/off status in every clock edges. The delay element is referred as tap hereafter. The clock for taking the snapshots is 500 MHz (2.0 ns), so if the number of taps is known where the input signal pulse went through during 2.0 ns, the delay amount per tap (dT) is known. Ignoring the statistical fluctutations, dT equials to "2000 (ps) / maximum tap number of reach", which is the time for the TDC 1-bit. However, each delay element in FPGA HR-TDCs has different tap delays; instead of the static calibration, a dynamic calibration that converts all tap numbers into time is required. Here, the tap number is called *fine-count*, and the value converted from fine count to the physical time is called *estimator*.

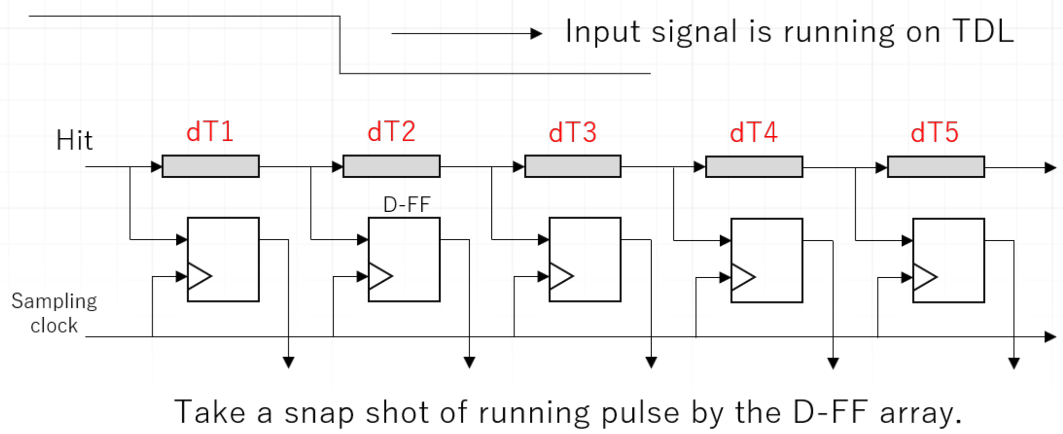

Figure 10: Schematics of Tapped-delay-line

Time calibration

Figure below shows the procedure for generating an estimator. First, a fine-count histogram has to be generated for a time-uncorrelated white noise spectrum as an input. The histogram weight for the each fine-count bin is proportional to the corresponding time delay of the bin. Once the histogram is ready, integrate the histogram counts of each bin up to the N-th to obtain the estimator; i.e. if the number of counts in the i-th bin is w_i, the N-th estimator is:

$$E_n = w_n/2 + \sum_{0}^{n-1}(w_i)$$

In the FPGA, required a mechanism to generat a fine-count histogram and a circuit to convert each hit from its fine-count to the estimator for output. As the signal source, the easiest is to use the detector signal to generate a fine count histogram. (The detector signal must be time-uncorrelated and random.) Histogram generation is independent of DAQ, and all input signals may be automatically filled into the histogram. However, this method requires a wait until the event accumulates to a fixed integra (e.g. 0x7ffff), and only available to the channel with a detector is connected. Therefore, a method to generate the histogram using a clock is prepared. A more practical method is to connect the calibration clock inside the FPGA to all input lines. This clock is adjusted to make a slight phase shift to the system clock which samples the TDL. In principle, the edge of the calibration clock can sweep the time in 1ps resolution. With the method of calibration clock, a histogram can be generated in tens of ms. It is assumed that the module is calibrated after the power is turned on, or at the beginning of each RUN.

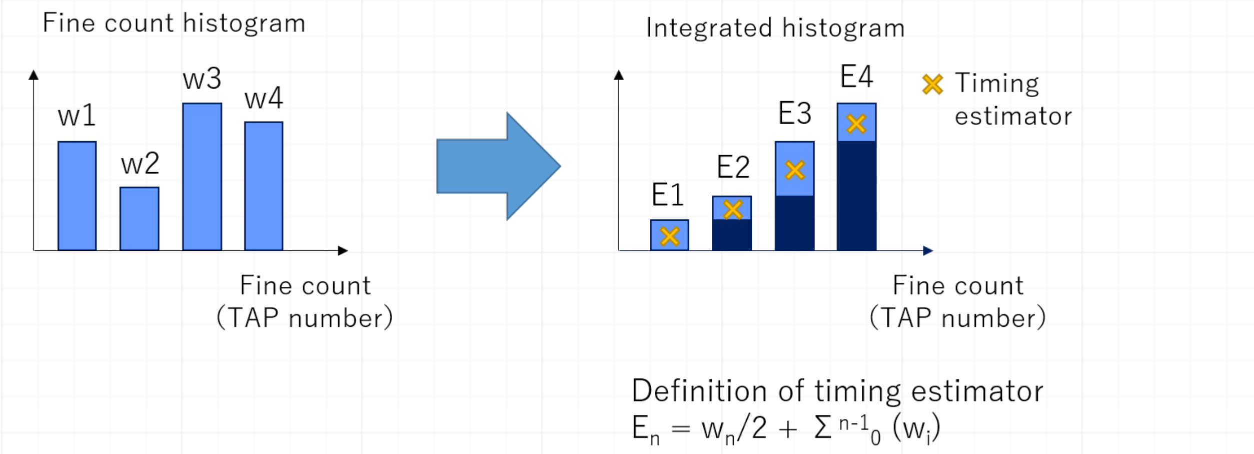

Figure 11: Procedure to generate estimators from fine count histogram.

Calibration block impremented and operation

The time calibration system is implemented by two RAMs. As shown in Figure, the fine-count is simultaneously fed into the two RAMs. One RAM generates a histogram. When the prefixed 0x7ffff event counts are accumulated, the RAM may be converted to the estimator mode, waiting for the swap. In the estimator mode, the RAM converts from a fine-count to an estimator. Since the raw estimator is 19 bits wide, which is too fine, and is discarded the lower 8 bits and make it 11 bits wide for output. The RAM swap is either automatic when one is ready, or manually switched. The behaviour is controled in `Controll::AutoSw` and `ReqSwitch` . If `Controll::AutoSw` is 1, it will switch automatically; if it is 0 and `ReqSwitch` is written, RAM will switch manually. Automatic switching is used when constant update of the RAM is necessary using the detector signal during the RUN.

Also, extraction of the fine-count without being converted to the estimator is possible. Set `Controll::Through` to 1 and the fine-count will appear directly.

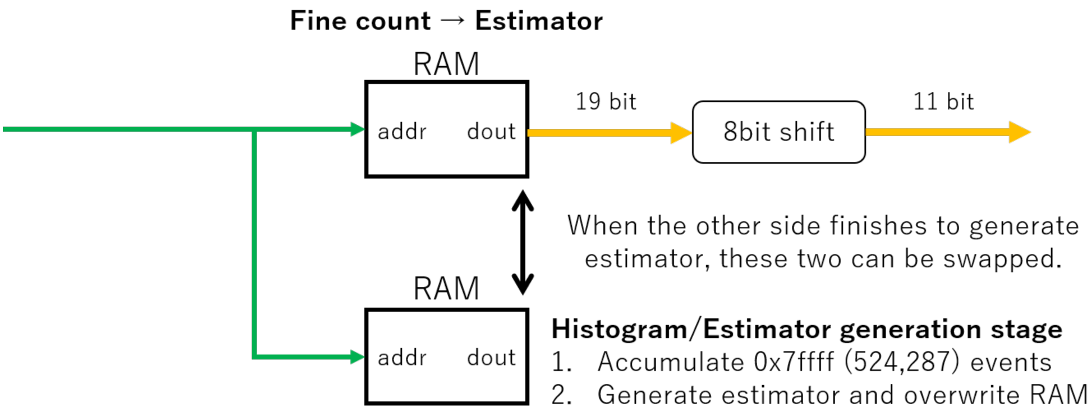

Figure 12: Swap pattern of Estimator look up table (LUT).

HR-TDC system

This section is a summary of the HR-TDC functions inside the Mezzanine HR-TDC. In this firmware, the length of TDL is 192 taps, which is rather too fine. Three taps are combined into one, conbined to 64 effective taps. Therefore, the maximum fine-count is 63. According to the measurements, approximately 55 taps are reached in 2.0 ns. The fine-count is passed to the clock area of 125 MHz, converted to the estimator, and then written to the ring buffer together with the hit bit. As shown in Figure, estimator (11bit) + semi-coarse count (2bit) + coarse count (11bit) gives a data length of 24bit. After that, the event is partially built in mezzanine HR-TDC and transferred to the HUL side. Approximately, Time (ns) = TDC value / 2048 / ClkFreq, where 2048 is the estimator's maximum and ClkFreq=0.500 GHz (FW v5.0 and later) or 0.520 GHz (up to FW v4.5) is the clock frequency. Use the TDC calibrator for a more accurate time conversion. The implementation after the Ring buffer is the same as in MH-TDC. When L1 trigger (common stop) is detected, the ring buffer is read and transfered. The range to search for hits may be set by `WindowMax` and `WindowMin` registers. These registers has the resolution of the 11-bit course count. Hits that do not fall within this range will not be written to the channel buffer. BUSY signal is asserted while searching for the hit information from the Ring buffer. Eventhough the structure is the same, the coarse count resolution is different from MH-TDC, since the system clock frequency is different.

The maximum number of hits for 1ch / event is placed when collecting data from channel buffer to block buffer. Curently, it is 16 hits, and data in the earlier time of TDC is discarted if more hits are recorded in the channel buffer, and the overflow bit is set.

Trigger output

Trigger signal can be output from HR-TDC from Version 5.0. The hit bits for all channels are logically summed and output. When trigger output is not used, it can be masked for each channel by TrigMask register.

High-resolution TDC specs	
TDC resolution	~30 ps
coarse count resolution	8.0 ns
Ring buffer length	16.3 us
timing resolution	20 ps (σ) *measured
minimum pulse width	~2 ns
double hit resolution	~4 ns
maximum hit/ch/event	16

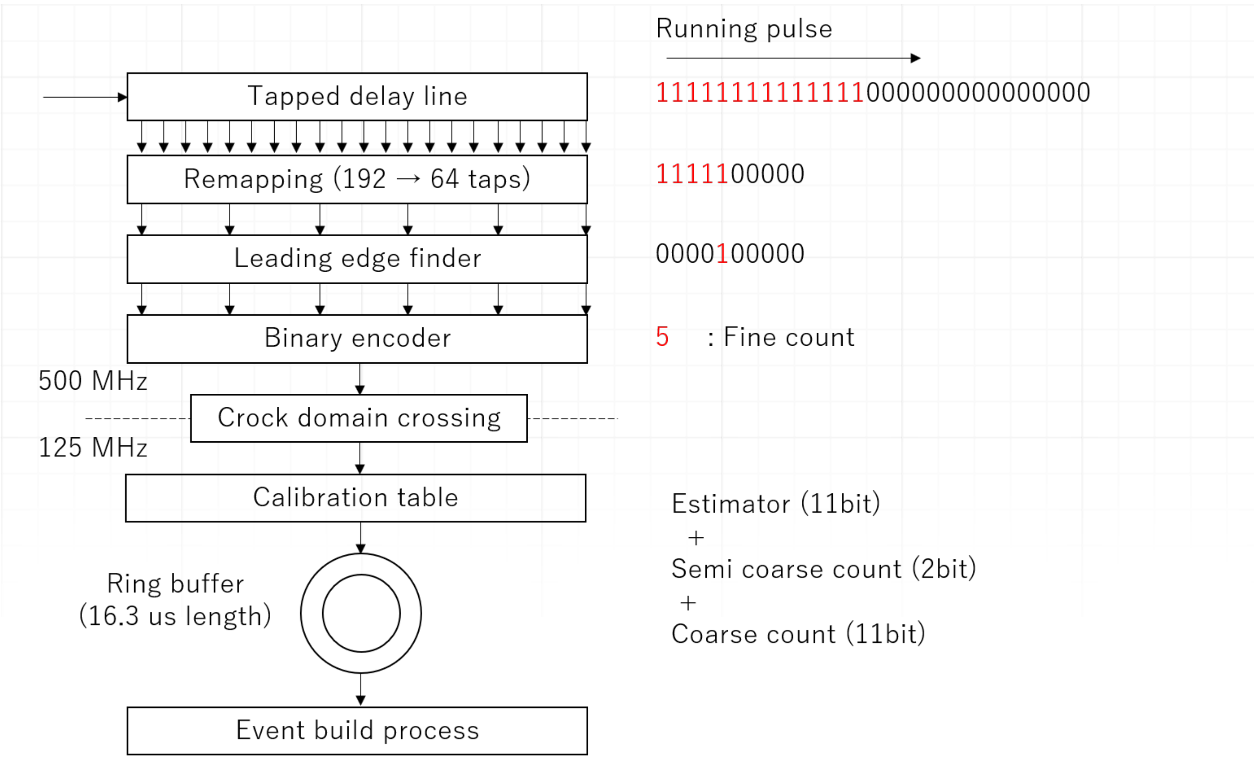

Figure 13: Block diagram of HR-TDC

4.8.2 Register map of Mezzanine HR-TDC

This section summarizes the registers of the Mezzanine HR-TDC. The registers described here belong to `namespace HRTDC_MZN` in `RegisterMap.hh`. Since the Mezzanine HR-TDC is accessed by bus bridging through BusBridge, it cannot be specified directly by the RBCP address. **Since Version 5.0, the address range has been expanded from 12-bit width to 16-bit width by modifying the local bus bridge.**

Register name	Address	Operation	Bit width	Description
Trigger Manager: DCT (module ID = 0x0)				
TestMode	0x0000	R/W	1	A mode that outputs a test pattern from the DDR transmitter to initialize the DDR receiver on the HUL side. Necessary for module initialization at power-on, used inside the distributed C++ software ddr_initialize.
ExtraPath	0x0010	R/W	1	When this bit is set, the signal input path switches from the input port to the calibration clock. Used to generate the LUT for the estimator with a calibration clock.
Gate	0x0020	R/W	1	DAQ gate. If it is 1, common stop is input to HR-TDC.
EnBlocks	0x0030	R/W	2	Enable the leading / trailing measurement block. The first bit is the leading block and the second bit is the trailing block. Since the default is 0, this bit must be set.
DAQ Controller: TDC (module ID = 0x01)				
Control	0x1010	R/W	3	A register for changing the operation of HR-TDC. The following three bits exist. Through (0x1) AutoSw (0x2) StopDout (0x4) If Through is 1, the fine-count is transferred without being converted to the estimator. If AutoSw is 1, the RAM will be swapped as soon as a new LUT is ready on the other RAM. If Stop Dout is 1, the stop data is also transferred as one word without subtracting from the common stop inside the FPGA.
ReqSwitch	0x1020	W	-	If AutoSw is 0, the estimator RAM will be swapped when this register is accessed. (Don't care about the register value.)
Status	0x1030	R	1	If this bit is 1, the next estimator LUT is ready on the alternative RAM.
PtrOfs	0x1040	R/W	11	Internal use. Do not touch.
WindowMax	0x1050	R/W	11	The upper limit of the time window to search for hits from the Ring buffer. 1bit is equivalent to 8.0 ns. See MH-TDC for details.

Register name	Address	Operation	Bit width	Description
WindowMin	0x1060	R/W	11	The lower limit of the time window to search for hits from the Ring buffer. 1bit is equivalent to 8.0 ns. See MH-TDC for details.
TrigMask	0x1070	R/W	32	Trigger output mask for each channel. Each bit corresponds to the channel. If 0 is set, the target channel is masked. E.g., if you want to mask channel 0, set 0xFFFF'FFFE.
DAQ Controller: SDS (module ID = 0xC)				
SdsStatus	0xC000	R	8	Obtains the status of SDS module.
XadcDrpMode	0xC010	R/W	1	DRP mode select for XADC. • 0x0: Read mode • 0x1: Write mode
XadcDrpAddr	0xC020	R/W	7	DRP address for XADC.
XadcDrpDin	0xC030	R/W	16	DRP input data for XADC
XadcDrpDout	0xC040	R	16	Obtains DRP output data from XADC.
XadcExecute	0xC050	W	-	Execute DRP access to XADC. (Don't care about the register value.)
SemCorCount	0xC0A0	R	16	Obtain the number of SEU corrected by SEM.
SemRstCorCount	0xC0B0	W	-	Reset SemCorCount to zero. (Don't care about the register value.)
SemErroAddr	0xC0C0	W	40	Address input for inject_address of SEM.
SemErroStrobe	0xC0D0	W	-	Sends pulse to inject_strobe of SEM. (Don't care about the register value.)
DAQ Controller: BCT (module ID = 0xE)				
Reset	0xE000	W	-	Asserts module reset signal from Bus Controller, and initializes all modules except SiTCP. (Don't care about the register value.)
Version	0xE010	R	32	Reads Firmware ID and versions. Multiple byte must be read out.
Reconfig	0xE020	W	-	Sends Low to PROG_B_ON to re-configure FPGA. SiTCP connection will be closed, and may be reconnected in a few seconds. (Don't care about the register value.)

4.8.3 Switch and LED on Mezzanine HR-TDC board

DIP SW functions

This function is assigned to the 4-bit DIP switch on the board.

switch number	function	details
1	-	Reserved
2	-	Reserved
3	-	Reserved
4	-	Reserved

LED function

The Mezzanine HR-TDC does not have a LED available to the user. The red LED lights when the FPGA is configured.

Module Busy timings

The definition of BUSY for Mezzanine HR-TDC is the OR of the BUSY signals listed below. In addition, BUSY of HUL HR-TDC BASE is also ORed. Normally, the BUSY length is equal to that for "Sequence busy".

BUSY type	BUSY length	description
Sequence busy	Depends on the search window width	Depends on WindowMax-WindowMin: BUSY while hits are searched in the Ring buffer
Block full	-	BUSY is output when the block buffer is full. It is asserted when the L1 trigger rate exceeds the data processing speed of the subsequent circuit. This happens when TCP transfer cannot catch up and is practically equivalent to SiTCP full.

4.8.4 Details of HUL HR-TDC BASE

HUL HR-TDC BASE has almost the same structure with MH-TDC except for DDR receiver and BusBridge. Unlike MH-TDC, HRM cannot be installed. Therefore, it cannot become a J0 bus master.

The DDR receiver needs to be initialized after the power is turned on; other than that there is no special action necessary. The BusBridge is provided for mezzanine slot U and D independently. BusBrige module bridges the BCT on the HUL side and the BCT on the mezzanine side. From the BCT on the HUL side, the BusBridge looks as a local module, and from the mezzanine side, it looks like an external link. Two actions are required to access the mezzanine. In the first action, the BCT on HUL stores the read/write command, the mezzanine local address, and the register value to `BBP::Txd`, a register located inside BusBrige. The second action sends signal to `BBP::Exec` for the brige action; BusBrige will start the communicating procedure with the mezzanine. Whether the BCT on the mezzanine side is driven by writing or by reading is determined by the command value specified in the first action. In the read mode, the value from the specified address will appear on `BBP::Rxd`. BusBrige will not respond to the BCT on HUL until the communication process is completed correctly. Therefore, the BCT on HUL will be deadlock, if `BBP::Exec` is called without a mezzanine HR-TDC card installed. If this occurs, `BCT::Reset` will not be called, and SiTCP Reset will be necessary. C++ functions for BusBrige control are grouped in `BctBusBridgeFunc.cc`. Details will be given in Chapter 6.

HR-TDC BASE ID and current version

ID	0x80eb
Mojor version	0x04
Minor version	0x01

Version history

Version	Release date	Changes
v1.5	2017.12.19	Initial release
v1.6	-	not released
v1.7	2018.02.02	Solved the issue that the event tag coming from the J0 bus was latched too early and the event number on HRM is deviated by 1. Repaired the issue that BCT hangs when calling BCT::Reset.
v3.7	2021.08.01	Added SDS and FMP. Installed Builder bus. Change of BCT structure.
v4.0	2023.01.17	Compatible with Mezzanine HR-TDC v5.0. Mezzanine HR-TDC v4.5 and earlier versions are not supported.
v4.1	2023.02.24	Bug-fixed version of HUL HRTDC BASE v4.0. Sometimes, v4.0 does not work correctly.

4.8.5 Register map of HUL HR-TDC BASE

The following is the register map of HUL HR-TDC BASE, defined in RegisterMap.hh as namespace HRTDC_BASE. Some of the names are identical with ones in Mezzanine; specify the namespace. Some registers are missing due to the un-support of HRM in IOM.

`RegisterMap.hh` for this firmware contains the global variables `kEnSlotup` and `kEnSlotDown` at the beginning. These are flags indicating which slot the mezzanine HR-TDC is installed. Set to false if Mezzanine HR-TDC is not installed. These are variables that are not necessary when the software is localized for a particular experiment.

Register name	Address	Operation	Bit width	Description
Trigger Manager: TRM (module ID = 0x00)				
SelectTrigger	0x0000'0000	R/W	12	Register to select Trigger source for TRM.
DAQ Controller: DCT (module ID = 0x01)				
DaqGate	0x1000'0000	R/W	1	ON/OFF DAQ gate. If DAQ gate is 0, TRM does not send trigger.
EvbReset	0x1010'0000	W	-	Write access to this address asserts a soft reset to Event Builder, and the self event counter is reset to zero. (Don't care about the register value.)
InitDDR	0x1020'0000	W	-	Initialization to DDR receiver. (Don't care about the register value.)
CtrlReg	0x1030'0000	R/W	4	Register to control DDR receiver. Details are described later.
Status	0x1040'0000	R	4	Status register of DDR receiver. Details are described later.
IO Manager: IOM (module ID = 0x02)				
NimOut1	0x2000'0000	R/W	4	Sets signal to NIMOUT1.
NimOut2	0x2010'0000	R/W	4	Sets signal to NIMOUT2.
NimOut3	0x2020'0000	R/W	4	Sets signal to NIMOUT3.
NimOut4	0x2030'0000	R/W	4	Sets signal to NIMOUT4.
ExtL1	0x2040'0000	R/W	3	Selects NIMIN for extL1
ExtL2	0x2050'0000	R/W	3	Selects NIMIN for extL2.
ExtClr	0x2060'0000	R/W	3	Selects NIMIN for Ext Clear
ExtBusy	0x2070'0000	R/W	3	Selects NIMIN for Ext Busy
cntRst	0x2090'0000	R/W	3	Selects NIMIN for coarse count reset of Mezzanine HR-TDC. Used to synchlonize multiple HR-TDCs.
Bus Bridge Primary: BBP (module ID = 0x3, 0x4)				
Txd	0x3000'0000	W	32	Data to be written to the slot-U secondary FPGA (FPGA on mezzanine card) via local bus bridge.
Rxd	0x3010'0000	R	32	Data read from the Slot-U secondary FPGA via the local bus bridge.
Exec	0x3100'0000	W	-	Assert the start signal to drive the Bus bridge primary and communicate with the slot-U secondary FPGA.
Txd	0x4000'0000	W	32	Data to be written to the slot-D secondary FPGA (FPGA on mezzanine card) via local bus bridge.
Rxd	0x4010'0000	R	32	Data read from the Slot-D secondary FPGA via the local bus bridge.

Register name	Address	Operation	Bit width	Description
Exec	0x4100'0000	W	-	Assert the start signal to drive the Bus bridge primary and communicate with the slot-D secondary FPGA.

Trigger Manager (TRM)

HUL HR-TDC BASE can not mount HRM; registers for HRM in TRM does not function.

Register name	Register value	Description
RegL1Ext	0x1	Select NIMIN as L1 trigger source.
RegL1J0	0x2	Select J0 bus as L1 trigger source
RegL2Ext	0x8	Select NIMIN as L2 trigger source.
RegL2J0	0x10	Select J0 bus as L2 trigger source
RegClrExt	0x40	Select NIMIN as Clear source
RegClrJ0	0x80	Select J0 bus as Clear source
RegEnL2	0x200	0: L2=L1 trigger、1: L2=L2入力
RegEnJ0	0x400	Use Tag info from J0 bus. Sends module busy to J0 bus if this bit is 1.

DAQ controller (DCT)

Details of `CtrlReg` and `Status` bits in DCT.

Register label	Bit number	Description
		CTRL Registers
RegTestModeU	1st bit (0x1)	If this bit is ON, test pattern receiving mode for initialization of DDR receiver (slot-U).
RegTestModeD	2nd bit (0x2)	If this bit is ON, test pattern receiving mode for initialization of DDR receiver (slot-D).
EnableU	3rd bit (0x4)	If this bit is ON, DDR receiver (slot-U) becomes available.
EnableD	4th bit (0x8)	If this bit is ON, DDR receiver (slot-D) becomes available.
FRstU	5th bit (0x10)	When this bit is set, it asserts a force reset signal to the FPGA on slot-U. A function implemented in the MIF block in firmware v3.7 and earlier.
FRstD	6th bit (0x20)	When this bit is set, it asserts a force reset signal to the FPGA on slot-D. A function implemented in the MIF block in firmware v3.7 and earlier.
		Status Registers
BitAlignedU	1st bit (0x1)	Data readout has become ready after finishing the bit slip in DDR receiver (slot-U).
BitAlignedD	2nd bit (0x2)	Data readout has become ready after finishing the bit slip in DDR receiver (slot-D).
BitErrorU	3rd bit (0x4)	Initialization failed; bit slip was tried for designated times on DDR receiver (slot-U), but not replied correctly.
BitErrorD	4th bit (0x8)	Initialization failed; bit slip was tried for designated times on DDR receiver (slot-D), but not replied correctly.

I/O Manager (IOM)

HRM is not supported and some of the registers do not function.

Register label	Register value	Description
		Signals available for NIMOUT
Reg_o_ModuleBusy	0x0	Module busy, not including J0 bus busy nor ExtBusy.
Reg_o_DaqGate	0x7	DAQ gate of DCT.
Reg_o_DIP8	0x8	DIP SW2 #8
Reg_o_clk1MHz	0x9	1 MHz clock
Reg_o_clk100kHz	0xA	100 kHz clock
Reg_o_clk10kHz	0xB	10 kHz clock
Reg_o_clk1kHz	0xC	1 kHz clock
Reg_o_TrigOutU	0xD	Assign the trigger output from mezzanine HR-TDC in Slot-U.
Reg_o_TrigOutD	0xE	Assign the trigger output from mezzanine HR-TDC in Slot-D.
Reg_o_TrigOutUD	0xF	Assigns the logical OR of the trigger output from both mezzanine HR-TDCs in Slot-U/D.
		NIMIN available for signals
Reg_i_nimin1	0x0	Assign to NIMIN1
Reg_i_nimin2	0x1	Assign to NIMIN2
Reg_i_nimin3	0x2	Assign to NIMIN3
Reg_i_nimin4	0x3	Assign to NIMIN4
Reg_i_default	0x7	If selected, Default setting are assgined to NIM IN/OUTs.

IOM Default registers.

NIM OUT	Register	
NIMOUT1	Reg_o_ModuleBusy	
NIMOUT2	reg_o_DaqGate	
NIMOUT3	reg_o_clk1kHz	
NIMOUT4	reg_o_DIP8	
Signal	**Register**	**Default**
ExtL1	Reg_i_Nimin1	NIMIN1
ExtL2	Reg_i_default	0
ExtLClear	Reg_i_default	0
ExtLBusy	Reg_i_nimin3	NIMIN3
cntRst	Reg_i_default	0

DIP SW2 functions

Lists functions assigned to DIP SW2.

Switch number	Function	Detail
1	SiTCP force default	ON for SiTCP forced default mode. Must be set before the power is turned on.
2	Not in use	
3	Force BUSY	Crate Busy and Module Busy turned to high. For connection check.
4	Bus BUSY	ON to include J0 bus busy in Crate Busy, OFF to otherwise.
5	LED	ON to light LED4.
6	Not in Use	
7	Not in Use	
8	Level	Reflects DIP8 on IOM

LED number	Description
LED1	Light for TCP open.
LED2	Light when module busy is high.
LED3	Light if DAQ gate is ON.
LED4	Light if DIP SW2 #5 (LED) is ON.

4.8.6 DAQ operation

Data flow is shown in Figure. There are two FPGAs involved from Mezzanine HR-TDC and BASE, however, the DAQ operation is the same with that in HUL MH-TDC. In each mezzanine, a partial event (up to the block buffer) is built and transferred to the BASE, where the received data is collected to built an event.

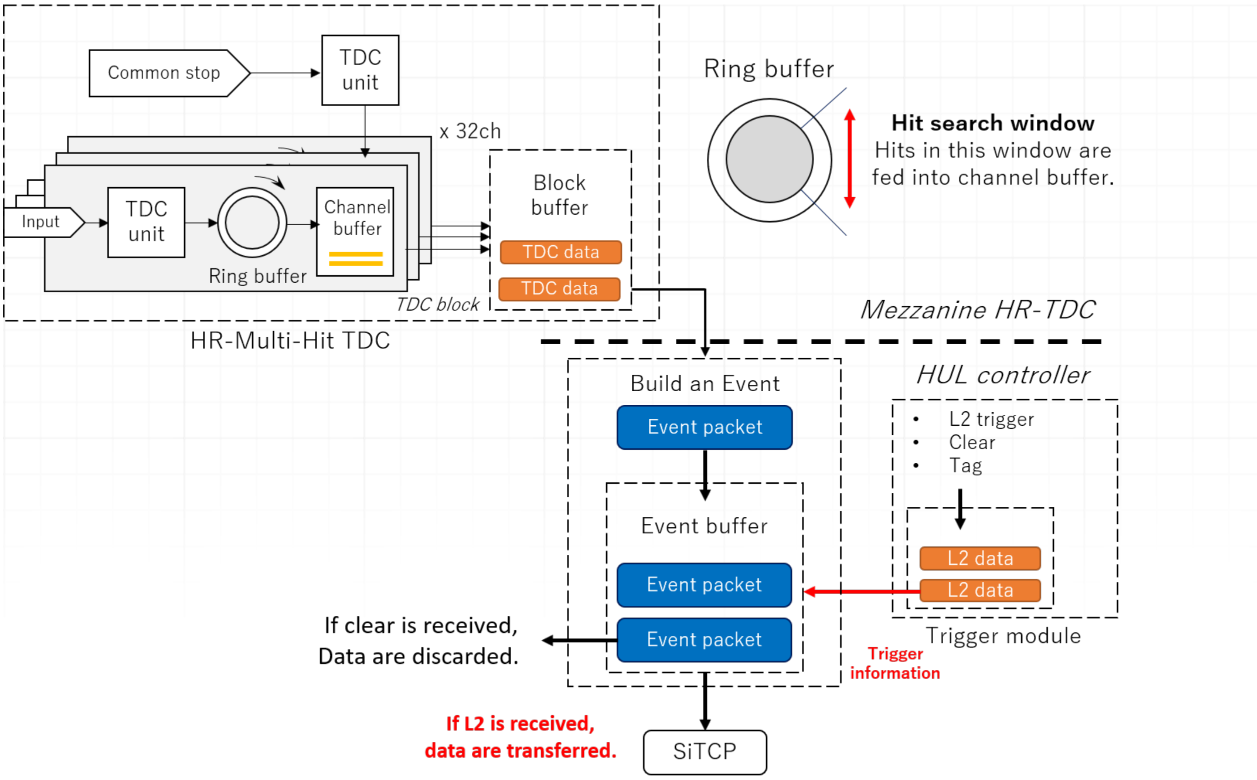

Figure 14: Data flow block diagram of Mezzanine HR-TDC and HUL HR-TDC BASE

Module Busy timing

Module busy is the OR of the followings:

BUSY type	BUSY length	Description
Self busy	210 ns	Asserted since the L1 trigger.
Mezzanine busy	Depends on the Mezzanine status	Busy from Mezzanine HR-TDC. Normally, the search window length of time.
Block full	-	BUSY when the block buffer of TDC Base becomes Full. The busy means that TCP transfer cannot catch up, and is practically equivalent to SiTCP full.
SiTCP full	-	TCP buffer of SiTCP becomes Full. It is asserted when the amount of data that the Event Builder is trying to send is large for the network bandwidth.

Data structure

Header word

```
Header1 (Magic word)
MSB                                                                          LSB
|                                  0xFFFF80EB                                  |

Header2 (event size)
|      0xFF00       |  OverFlow  |     "000"     |        Number of word (12-bit)                |
```

Number of word indicates the number of words contained in the data body. It includes the two words for Sub-header. So the lowest value is 2. OverFlow stands when there is an over flow channel even for 1ch in the entire HUL.

```
Header3 (event number)
|   0xFF    | "0000" |  Tag (4-bit)  |                 Self counter (16-bit)                |
```

Tag is the 4-bit Tag information from TRM. The lower 3 bits are the lower 3 bits of the RM Event Number, and the 4th bit is the least significant bit of the RM spill number. Self-counter is a local event number that is incremented each time an event is forwarded. Starts with 0.

```
Sub-headers
|  0xFA00  | "0" |  OverFlow  |  StopDout  |  Through  |          # of word (12-bit)          |
|  0xFB00  | "0" |  OverFlow  |  StopDout  |  Through  |          # of word (12-bit)          |
```

Header of mezzanine HR-TDC. `0xFA00` and `0xFB00` correspond to slot -U and -D, respectively. Overflow indicates the presence of over flow in each mezzanine. StopDout and Through indicate the state of `Stop Dout` and `Through` in `HRTDC_MZN::TDC::Controll`, respectively. "Number of word" indicates the number of words for each mezzanine.

Data body

```
Data body
|  Magic word (3-bit)  |  Ch (5-bit)  |                      TDC (24-bit)                    |
```

The *Magic word* is defined as follows.

- 6 Leading
- 5 Trailing
- 4 Common stop

Ch can only be counted up to 31ch as it is 5 bit width. Check if the data belongs to subheaders A or B, decode it, and if it belongs to subheader B, add 32 channels. As mentioned in the section for HR-TDC, *TDC* is estimator (11bit) + semi-coarse count (2bit) + coarse count (11bit), which makes a total of 24 bits. When *Through* is ON, the fine-count appears at the estimator position.

4.9 Three-dimensional matrix trigger

このファームウェアのバージョン表記は他のFWと異なっており、メジャーバージョンが1ですがFMPやSDSは搭載されています。

固有名	0xe033
メジャーバージョン	0x01
マイナーバージョン	0x01

更新歴

バージョン	リリース日	変更点
v1.1	2020.12.02	実験で利用した最終版

動作概要

このファームウェアはJ-PARC E03実験とE42実験の際に用いられたマトリックスコインシデンストリガーです。似たような機能を持つマトリックストリガー回路を作成したい場合の例題として利用してください。本ファームウェアにはデータ収集機能はありません。このFWでは3種類のホドスコープの三次元相関からトリガーを生成します。図にブロック図を示します。この実験ではBH2、TOF、SCHという3種類のタイミングホドスコープ間のマトリックス相関を用います。丸カッコ内の数字はチャンネル番号を表しており、合計14336パターン (8x28x64) の組み合わせが発生します。入力は二重FFで同期されます。クロック速度は350 MHzです。後述のDWGやマトリックスパターンのブロックも同様のクロックで駆動されています。BH2がNIM-INへも繋がれているのはテストを簡便に行うためです。このファームウェアはメザニンスロットがSCHの入力を受けるためDCR v1/v2が必須です。

同期された入力信号はDelay Width Generator (DWG) で幅と遅延時間の調整がされます。各DWGはRBCPを通じて調整が可能です。350 MHz (2.857... ns) のクロック精度で32段階の調整が可能です。詳しくはソフトウェアの項で述べます。DWGではパルス出力中にもう一度パルス入力があった場合2つのパルスを繋げます。麻痺型モデルで表されるデッドタイムの振る舞いと同様です。

MTX3DとMTX2Dはそれぞれ三次元マトリックスコインシデンスと二次元マトリックスコインシデンスのブロックです。このFWでは1つ三次元トリガーと2つの二次元トリガーが実装されており、それぞれマトリックスパターンを設定可能です。RBCPを用いて全てのマトリックスエレメント1つ1つのOn/Offの切り替えが可能です。どのように実現しているかは後述します。

各マトリックストリガーはIOMを通じてNIMOUTから出力可能です。このFWでは実験の要求から二次元マトリックスの出力を三次元マトリックスの結果でVETOする経路が用意されています。この実験では三次元トリガーがビームVETOの役割を果たしていたためタイミングが良く分かっており、二次元トリガーに対しては固定長ディレイでVETO位置を調整しています。このあたりは実験条件に合わせて変更してください。

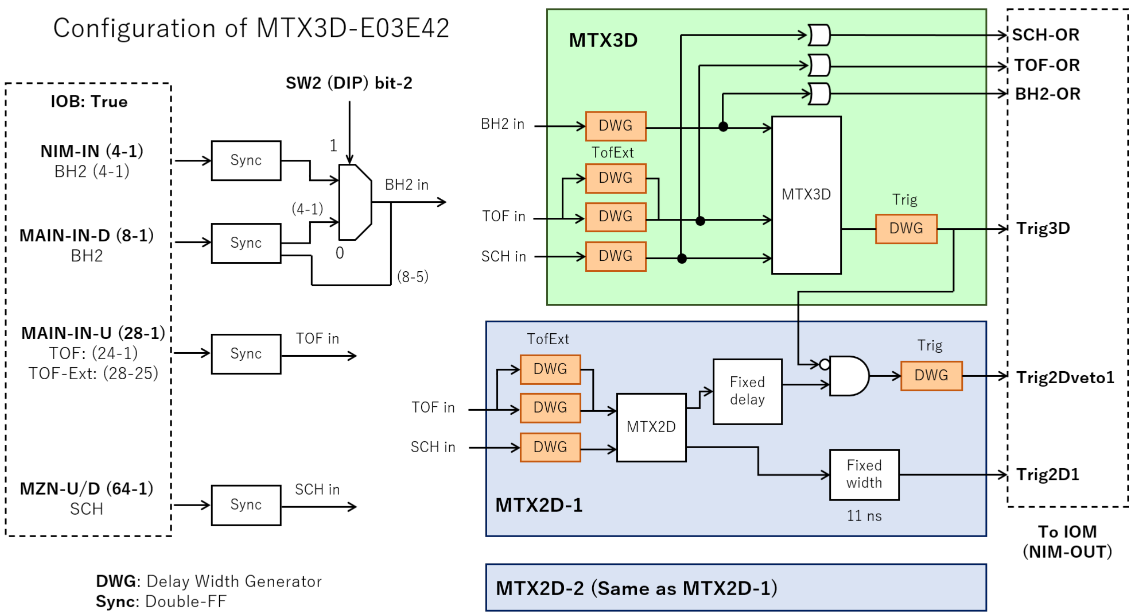

Figure 15: Matrix3D triggerのブロック図。左側入力、右側が出力。

4.9.1 マトリックスパターンの設定方法

14336パターンのスイッチをアドレスで解決しようと思うととてつもなく大きなエンコーダが必要になり、一般的にFPGAで実現するのは非現実的です。ここで3次元のレジスタを(28x64)x8と分解すると、8-bit幅・1792長のレジスタと捉えることが出来ます。すなわち、8-bit幅で長さが1792のシフトレジスタを用意することで、全てのレジスタビットの設定が可能となります。二次元の場合1-bit幅で長さが1792のシフトレジスタを用意します。RBCP（BCT）は同一のアドレスに1792回書き込むだけで設定が実現でき、極めてリソース効率が良いです。レジスタ設定はシステムクロックに対して低速なクロックで行っても良いため、このFWでは10 MHzのクロックでシフトレジスタを駆動しています。

4.9.2 DWGの構造

DWGは遅延と幅の生成をシフトレジスタへのビットパターンのプリセットによって実現しています。例えば先頭から`00011111000...`というビットパターンを入力があったタイミングでシフトレジスタへセットしたとします。そうするとこのパターンは遅延量が3でパルス幅が5の波形を与えます。遅延量と幅の情報からビットパターンへの変換はソフトで行う事にしています。最大幅と遅延量がそれぞれ32のため、DWGの取るレジスタ幅は64-bitです。

4.9.3 レジスタマップ

レジスタ名	アドレス	読み書き	ビット幅	備考
		Trigger Manager: MTX2D-1 (module ID = 0x00)		
TofDWG	0x0000'0000	R/W	64	Tof (1-24) 入力のDWGレジスタを設定します。
TofExtDWG	0x0010'0000	R/W	64	Tof (25-28) 入力のDWGレジスタを設定します。
SchDWG	0x0020'0000	R/W	64	SCH入力のDWGレジスタを設定します。
TrigDWG	0x0030'0000	R/W	64	二次元トリガー出力のDWGを設定します。
EnableMtx	0x0040'0000	W	1	二次元マトリックスパターンを設定するアドレスです。シフトレジスタの最下位にレジスタを書きます。
ExecSR	0x0100'0000	W	-	二次元マトリックスパターンを設定するシフトレジスタを1つシフトさせます。
		Trigger Manager: MTX2D-2 (module ID = 0x01)		
		各レジスタはMTX2D-1と同様です		
		Trigger Manager: MTX3D (module ID = 0x02)		
TofDWG	0x2000'0000	R/W	64	Tof (1-24) 入力のDWGレジスタを設定します。
TofExtDWG	0x2010'0000	R/W	64	Tof (25-28) 入力のDWGレジスタを設定します。
SchDWG	0x2020'0000	R/W	64	SCH入力のDWGレジスタを設定します。
Bh2DWG	0x2030'0000	R/W	64	SCH入力のDWGレジスタを設定します。
TrigDWG	0x2040'0000	R/W	64	三次元トリガー出力のDWGを設定します。
EnableMtx	0x2050'0000	W	8	三次元マトリックスパターンを設定するアドレスです。シフトレジスタの最下位にレジスタを書きます。
ExecSR	0x2100'0000	W	-	三次元マトリックスパターンを設定するシフトレジスタを1つシフトさせます。
		Trigger Manager: IOM (module ID = 0x03)		
Nimout1	0x3000'0000	R/W	4	Nimout1ポートへの出力を設定します。
Nimout2	0x3010'0000	R/W	4	Nimout2ポートへの出力を設定します。
Nimout3	0x3020'0000	R/W	4	Nimout3ポートへの出力を設定します。
Nimout4	0x3030'0000	R/W	4	Nimout4ポートへの出力を設定します。

I/O Manager (IOM)

レジスタラベル	レジスタ値	備考
		NIMOUTへ出力可能な内部信号
Reg_o_Trig3D	0x0	MTX3Dトリガーを出力します。
Reg_o_Trig2DVeto1	0x1	MTX3DでVETOされた後のMTX2D-1トリガーを出力します。
Reg_o_Trig2DVeto2	0x2	MTX3DでVETOされた後のMTX2D-2トリガーを出力します。
Reg_o_Trig2D1	0x3	MTX2D-1トリガーを出力します。
Reg_o_Trig2D2	0x4	MTX2D-2トリガーを出力します。
Reg_o_TofOR	0x5	TofORを出力します。
Reg_o_SchOR	0x6	SchORを出力します。
Reg_o_Bh2OR	0x7	Bh2ORを出力します。

4.10 Mass trigger (TOF based trigger)

このファームウェアは2023年現在更新がストップしています。Mezzanine HR-TDC v5.0や最新のソフトウェアとは互換性がありません。以前のバージョンの物を引き続き利用してください。

固有名	0x20d1
メジャーバージョン	0x03
マイナーバージョン	0x00

更新歴

バージョン	リリース日	変更点
v3.0	2020.12.02	実験で利用した最終版

Mass trigger (MsT) はJ-PARC K1.8での通称であり、機能的にはTOFベースのトリガー生成回路です。ダイポール磁気スペクトロメータを通過した粒子の軌道は大雑把に粒子の運動量と相関があります。2つのホドスコープの二次元マトリックス相関から運動量範囲を制限し、各マトリックスエレメントに対してTOFの分布を取ると粒子の質量ごとにTOFピークが分離します（低い運動量であれば）。Mass triggerはmezzanine HR-TDCとHULのメイン入力を用いて、二次元マトリックスとTOF情報によるトリガー生成を行うためのファームウェアです。HR-TDCの情報を得るためにはcommon stop入力が必要であるため、mass triggerはlevel2 decisionを行い、level2 triggerかもしくはfast clear信号を生成することが仕事です。

ブロック図を図に示します。MsTはHRMとHR-TDCのメザニンカードを必要としています。それぞれ、slot-Uとslot-Dへマウントしてください。Main INへ入力された信号は低速なTDCでデジタイズされヒットレジスタ情報を与えます。そのため、このファームウェアにおけるTOFは厳密には検出器間の時間差でなく、common stopとの時間差です。Mass triggerを利用するためにはlevel1 triggerが非同期回路で生成されている必要があります。この仕様はK1.8ビームラインの都合に合わせてあるため、真のTOFを計算させたい場合ファームウェアの改修が必要になります。

SCH (64ch)とTOF (32ch) 間の2048パターンに対してTOF windowのチェックを行います。この時、HR-TDCから返ってきたマルチヒットデータ全てに対して判定を行うため、同一チャンネルに対して複数回の判定が行われることがあります。1つでもアクセプト範囲にTOF値があればトリガーが発行されます。1つのアクセプト範囲にデータがない場合クリア信号が出力されます。アクセプト窓の設定はSiTCP経由で行います。MsTでトリガー判定を行いたいのはlevel1 triggerが物理トリガーの場合だけです。キャリブレーショントリガーの場合は判定無しでlevel2 triggerを発生させなければいけません。判定すべきlevel1 triggerかどうかを知らせるために、このファームウェアではpiK flagという信号をlevel1 triggerに続いて入力することになっています。Flag入力が無ければ判定を行わず、必ずlevel2 triggerが発行されます。

本ファームウェアにはデータ収集機能が存在します。HR-TDCとLR-TDCによって得られた時間情報と、MsTの判定結果がデータとして返ってきます。本ファームウェアはトリガー生成回路ですが、データ収集のためには外部からトリガー入力が必要です。

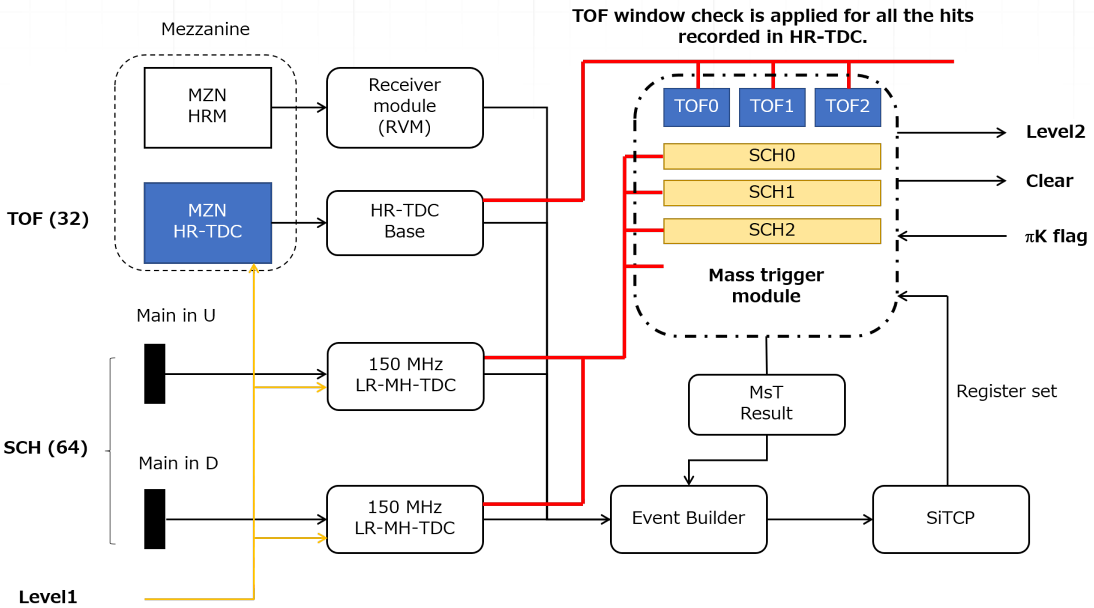

Figure 16: Mass triggerのブロック図。

タイミングチャート

K1.8ビームラインでmass triggerを導入する目的は、主にVMEモジュール用にfast clearを生成し不必要なバスアクセス時間を減らす事です。K1.8ビームラインではchained-block-transfer (CBLT) でVMEバスアクセスを行っていますが、おおよそ100 usのバスアクセス時間がかかります。VMEモジュール内のmulti-event bufferを活用して、バスアクセス時間をlevel1 triggerに対するbusyに含めない工夫はしていますが、潜在的なbusyであるためlevel1 triggerのレートが7 kHzあたりから急激にDAQ効率が悪くなります。そのような実験ではmass triggerを導入してlevel2判定を加えてバスアクセスの回数を減らします。

図に想定タイミングチャートを示します。Mass triggerはHR-TDCとLR-TDCにcommon stop入力が入った時点から動作を開始します。HR-TDCからデータの転送が終わると判定を開始します。Mass triggerはmulit-hitを全て処理するため、ここまでの時間は可変です。Level1から固定時間後に判定結果を出力するために `MST::TimerPreset` で出力までの時間を設定します。今のファームウェアバージョンでは500程度の大きな値を設定することを推奨しています。 `MST::TimerPreset` は判定プロセスよりも優先度が高いため、指定時間後に判定が終わっていなかった場合後述のno decision flagを立てたうえで強制的にlevel2 triggerを出力します。No decision flagが頻繁に立つ場合 `MST::TimerPreset` の値が小さすぎます。

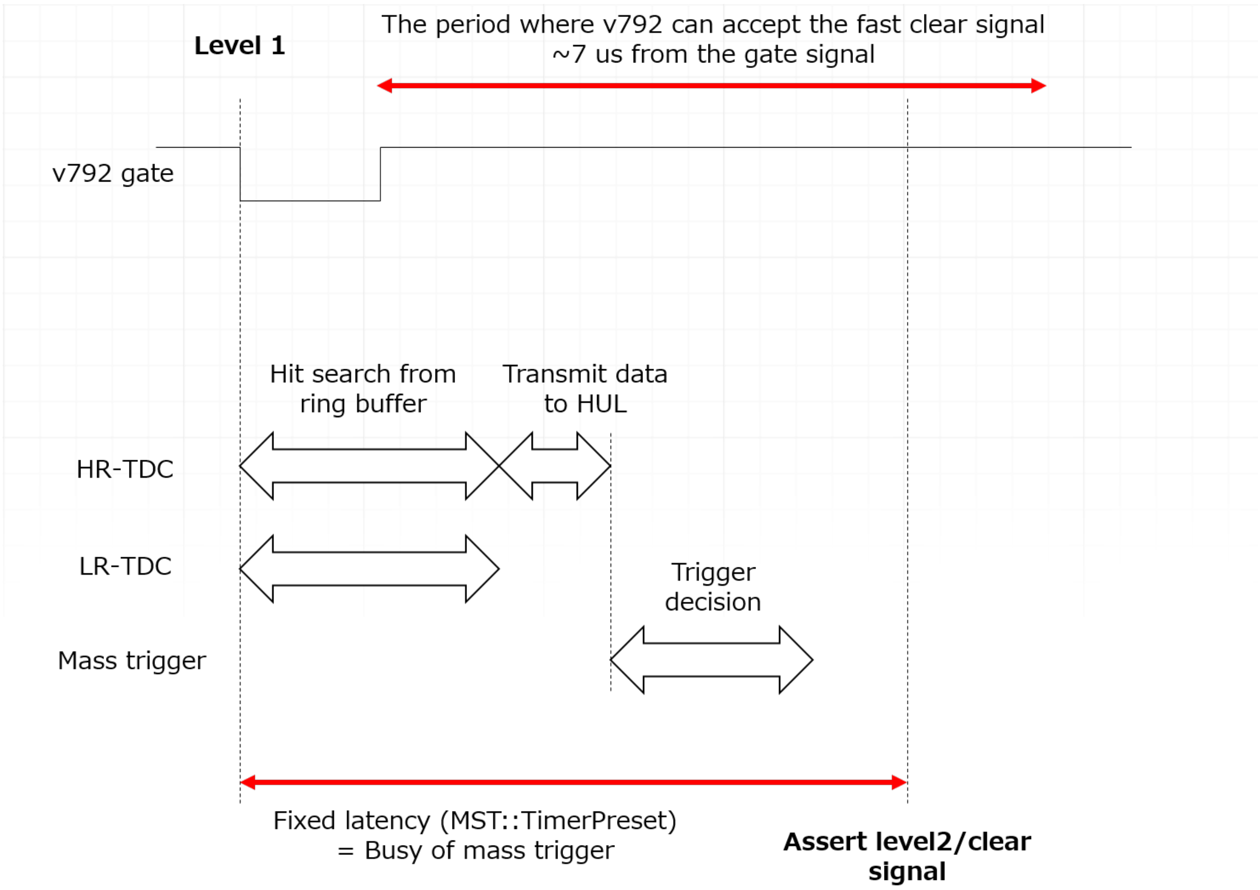

Figure 17: 想定しているlevel1 triggerからlevel2判定までのタイムチャート。

判定フロー

Level2 trigger判定のフロー図を図に示します。図の下側に書かれているリストは各判定状態におけるフラグ（および内部信号）の状況を示しています。判定動作はlevel1 trigger受信で開始します。TDCブロックから情報を集め判定回路が各チャンネル独立に動作します。 `MST::TimerPreset` に指定した時間がたった後、piK flagを受信しているか、および判定プロセスがすべて終了しているかのチェックを行います。どちらかを満たしていない場合、no decision flagを立ててlevel2 triggerを出力します。次にHR-TDCから取得したTDC値がアクセプト範囲にあるかどうかのチェックを行います。1つでも存在すればmass trigger acceptととなり、level2 triggerが出力されます。1つもない場合clear判定となり、敗者復活判定へ続きます。Clear判定を下されたイベントはDAQで取得されないため、クリアしているイベントをサンプル検査するために `MST::ClearPreset` に指定した値に1度、敗者復活アクセプトを出します。敗者復活判定が下された場合consolation acceptのフラグが立ち、level2 triggerが出力されます。敗者復活の条件を満たさない場合fast clearが出力されます。

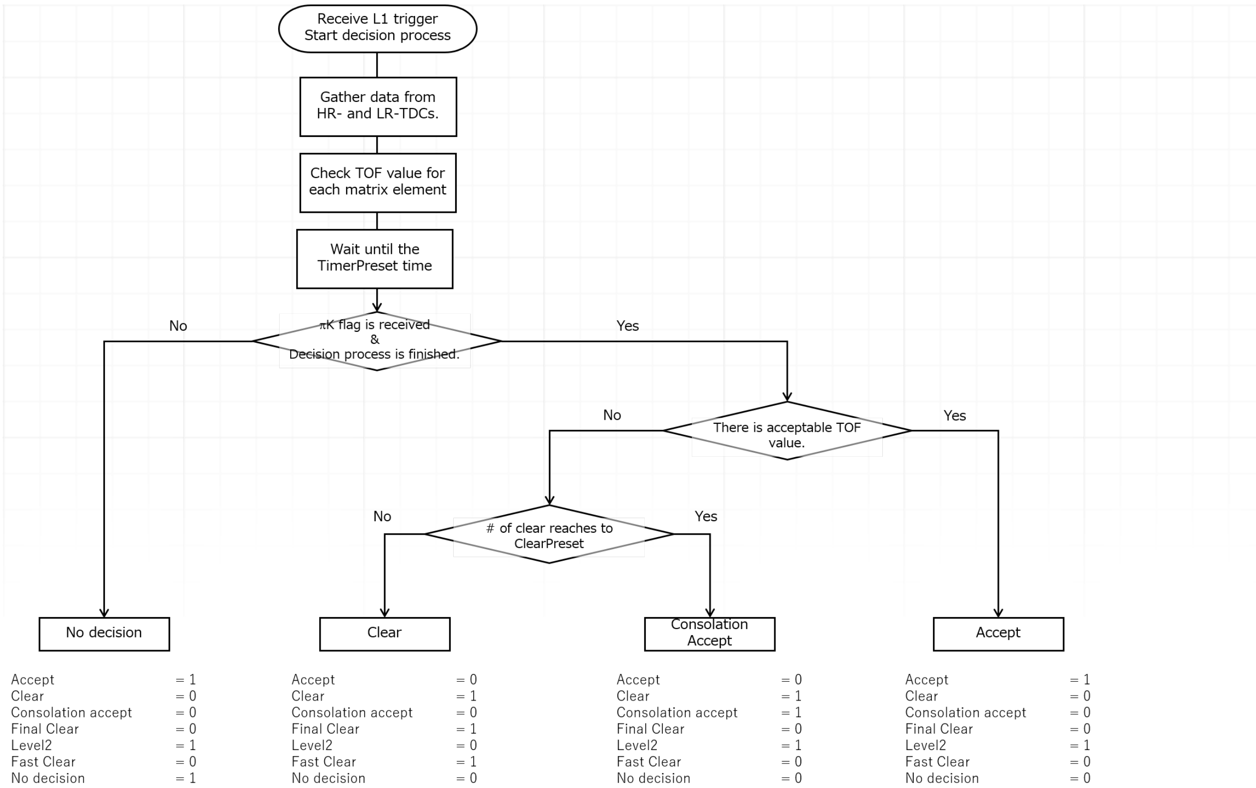

Figure 18: Mass triggerの判定フロー。

TOFのアクセプト範囲の設定

Mass triggerには2種類のTDC範囲設定を行うレジスタが存在します。`LRTDC::WinMax`と`LRTDC::WinMin`、および mezzanine HR-TDCのウィンドウレジスタはcommon stop入力に対してヒットサーチを行う範囲を設定します。これは通常のTDCファームウェアと同様です。`MST::WixMax`と`MST::WinMin`はlevel2判定でアクセプトとなる範囲を与えます。Mass triggerでは32 chのHR-TDC入力と64 chのLR-TDC入力間の二次元マトリックスに対して、行列要素毎にこのレンジを設定することが出来ます。マトリックスコインシデンストリガーの時と同様に24-bit幅で長さが2048のシフトレジスタを採用しています。`MST::WixMax`と`MST::WinMin`の両方を0に設定すると、その行列要素に対しては判定を行いません。

4.10.1 レジスタマップ

レジスタ名	アドレス	読み書き	ビット幅	備考
Trigger Manager: TRM (module ID = 0x00)				
SelectTrigger	0x0000'0000	R/W	12	TRM内部のトリガーポートの選択を行うレジスタ。
DAQ Controller: DCT (module ID = 0x01)				
DaqGate	0x1000'0000	W	-	DAQ gateのON/OFF。DAQ gateが0だとTRMはtriggerを出力できない。
EvbReset	0x1010'0000	R/W	1	このアドレスへ書き込み要求することでEVBのソフトリセットがアサートされ、Event builder内部のセルフイベントカウンタが0になる。
InitDDR	0x1020'0000	W	-	DDR receiverへ向けて初期化要求を行う。
CtrlReg	0x1030'0000	R/W	4	DDR receiverを制御するためのレジスタ。
Status	0x1040'0000	R	4	DDR receiverのステータスレジスタ。
IO Manager: IOM (module ID = 0x02)				
NimOut1	0x2000'0000	R/W	5	NIMOUT1へ何を出力するかを設定する。
NimOut2	0x2010'0000	R/W	5	NIMOUT2へ何を出力するかを設定する。
NimOut3	0x2020'0000	R/W	5	NIMOUT3へ何を出力するかを設定する。
NimOut4	0x2030'0000	R/W	5	NIMOUT4へ何を出力するかを設定する。
ExtL1	0x2040'0000	R/W	3	extL1にどのNIMINを接続するか設定。
ExtL2	0x2050'0000	R/W	3	extL2にどのNIMINを接続するか設定。
ExtClr	0x2060'0000	R/W	3	Ext clearにどのNIMINを接続するか設定。
ExtBusy	0x2070'0000	R/W	3	Ext busy入力にどのNIMINを接続するか設定。
cntRst	0x2090'0000	R/W	3	Mezzanine HR-TDC内部のcoarse countをリセットするハードリセット信号。複数台のHR-TDCを同期したい場合に使用する。この線にどのNIMINを接続するか設定。
PiKTrig	0x20A0'0000	R/W	3	Mass triggerに対する物理トリガーフラグ入力。Level1 triggerに続いて入力があるとlevel2判定を行う。
IO Manager: MIF-Down				
Connect	0x3000'0000	R/W	-	MIF-Downからmezzanine HR-TDCのBCTへ向けて通信プロセスを開始する。アクセスする際のモードが書き込みなのか読み出しなのかで、メザニンのBCTへのアクセス方法が切り替わる仕様となっている。
Reg	0x3010'0000	R/W	20	MIF-Downにmezzanine HR-TDC用のlocal addressと書き込み用レジスタ値をMIFに一時的に保存する。 • [19:8]: Local address • [7:0]: Register valus
ForceReset	0x3100'0000	W	-	MIF-Downのmezzanine HR-TDCへ強制リセット信号をアサート。DAQやBCTがハングした場合に使用する。
Low-resolution TDC: LRTDC				
PtrOfs	0x4010'0000	R/W	11	内部制御変数。ユーザーは触らない。

レジスタ名	アドレス	読み書き	ビット幅	備考
WindowMax	0x4020'0000	R/W	11	Ring bufferからヒットを探す時間窓の上限値。1-bitが6.666... nsに相当。
WindowMin	0x4030'0000	R/W	11	Ring bufferからヒットを探す時間窓の下限値。1-bitが6.666... nsに相当。
		Mass trigger: MST		
ClearPreset	0x5000'0000	R/W	7	このレジスタに書かれた値に1度、敗者復活アクセプトが出力される。0に設定すると敗者復活判定を行わない。
TimerPreset	0x5010'0000	R/W	9	Level1 triggerを受信してからlevel2 trigger/clearを出力するまでの遅延時間。
WinMax	0x5020'0000	W	24	アクセプトするTOF範囲の上限値を与えます。シフトレジスタの最下位にレジスタを書きます。
WinMin	0x5030'0000	W	24	アクセプトするTOF範囲の下限値を与えます。シフトレジスタの最下位にレジスタを書きます。
Exec	0x5040'0000	W	-	シフトレジスタを1つシフトさせます。
Bypass	0x5050'0000	R/W	1	このレジスタが1になると判定回路をバイパスしてlevel2 triggerを出力するようになります。実験中に一時的にmass triggerをバイパスしたい場合に利用します。

I/O Manager (IOM)

レジスタラベル	レジスタ値	備考
		NIMOUTへ出力可能な内部信号
Reg_o_ModuleBusy	0x0	Module busyです。Module busyは自身の内部busyのみを指します。J0 busのbusyやExtBusyは含まれません。
Reg_o_CrateBusy	0x1	CrateBusyです。CrateBusyはmodule busyに加えてJ0 busのbusyやExtBusyを含みます。J0 busマスタの場合に利用する信号になり、またHRMが Trigger Moduleへ返すbusyと同等です。
Reg_o_RML1	0x2	HRMが受信したL1 triggerを出力します。
Reg_o_RML2	0x3	HRMが受信したL2 triggerを出力します。
Reg_o_RMClr	0x4	HRMが受信したL2 triggerを出力します。
Reg_o_RMRsv1	0x5	HRMが受信したReserve 1を出力します。
Reg_o_RMSnInc	0x6	HRMがSpill Number Incrementを出力します。
Reg_o_DaqGate	0x7	DCTのDAQ gateを出力します。
Reg_o_DIP8	0x8	DIP SW2 8番のレベルを出力します。
Reg_o_clk1MHz	0x9	1 MHzのクロックを出力します。
Reg_o_clk100kHz	0xA	100 kHzのクロックを出力します。
Reg_o_clk10kHz	0xB	10 kHzのクロックを出力します。
Reg_o_clk1kHz	0xC	1 kHzのクロックを出力します。
Reg_o_clk1kHz	0xC	1 kHzのクロックを出力します。
Reg_o_Accept	0xD	Accept flagを出力します。
Reg_o_Clear	0xE	Clear flagを出力します。
Reg_o_ConsolationAccept	0xF	Consolation accept flagを出力します。
Reg_o_FinalClear	0x10	Final clear flagを出力します。
Reg_o_Level2	0x11	Level2 flagを出力します。
Reg_o_FastClear	0x12	Fast clear flagを出力します。
Reg_o_NoDecision	0x13	No decision flagを出力します。
		内部信号線へ割り当て可能なNIMINポート
Reg_i_nimin1	0x0	NIMIN1番を信号線へアサインします。
Reg_i_nimin2	0x1	NIMIN2番を信号線へアサインします。
Reg_i_nimin3	0x2	NIMIN3番を信号線へアサインします。
Reg_i_nimin4	0x3	NIMIN4番を信号線へアサインします。
Reg_i_default	0x7	このレジスタが設定された場合、指定のデフォルト値がそれぞれの内部信号線へ代入されます。

以下はIOMレジスタの初期値のテーブルです。

NIM出力ポート	初期レジスタ	
NIMOUT1	Reg_o_ModuleBusy	
NIMOUT2	reg_o_RML1	
NIMOUT3	reg_o_RML2	
NIMOUT4	reg_o_RMClr	
内部信号線	**初期レジスタ**	**デフォルト値**
ExtL1	Reg_i_Nimin1	NIMIN1
ExtL2	Reg_i_default	0
ExtLClear	Reg_i_default	0
ExtLBusy	Reg_i_nimin3	NIMIN3
ExtLRsv2	Reg_i_nimin4	NIMIN4
cntRst	Reg_i_default	0
PiKFlag	Reg_i_nimin2	NIMIN2

4.10.2 HUL上のスイッチ・LEDの機能

DIP SW2の機能

HUL RMと同様です。

LED点灯の機能

HUL RMと同様です。

4.10.3 DAQの動作

Mass triggerからはLR-TDCとHR-TDCの測定結果、およびlevel2判定の結果（フラグ）が返ってきます。Level2判定はイベント毎に行うため、level2判定を行っている間BUSYになります。

Module Busyとなるタイミング

BUSYの定義は以下に列挙するBUSY信号のORになります。

BUSY種別	BUSY長	備考
Self busy	210 ns	L1 triggerを検出した瞬間から固定長でアサート。
Sequence busy	サーチ窓幅に依存	Ring bufferからヒット情報を探している間、すなわち`WindowMax`-`WindowMin`分のBUSYが出力されます。Mass triggerにはLR-TDCとHR-TDCが存在するため、長いほうが採用されます。
Block full	-	ブロックバッファがfullになった段階でBUSYが出力されます。L1 triggerレートが後段の回路のデータ処理速度を上回るとアサートされます。つまりTCP転送が追いつかない事を意味するので、実質的にSiTCP fullと同等です。
SiTCP full	-	SiTCPのTCPバッファがFullになると出力されます。ネットワーク帯域に対してEvent Builderが送信しようとするデータ量が多いとアサートされます。
Decision busy	`MST::TimerPreset`に依存	Level1 trigger受信からlevel2/clear出力までの間BUSYになります。`MST::TimerPreset`に依存します。

データ構造

ヘッダワード

```
Header1 (Magic word)
MSB                                                                                  LSB
|                                         0xFFFF20D1                                   |

Header2 (event size)
|   0xFF    | "00" |  OerFlow  |    "000"    |         Number of word (12-bit)            |
```

Number of wordはデータボディに含まれるワード数を示します。Number of WordはSub-header分の2ワード分を含みます。なので最低値が2です。Over flowはHUL全体で1chでもover flowチャンネルがあると立ちます。

```
Header3 (event number)
|   0xFF    | HRM bit | "000" |  Tag (4-bit)  |             Self counter (16-bit)            |
```

TagはTRMから出力される4bitのTag情報です。下位3bitがRM Event Numberの下位3ビット、4ビット目がRM spill numberの最下位ビットとなります。Self-counterはイベント転送を行うたびにインクリメントされるlocal event numberで、0オリジンです。

```
Data body
RVM data
|  0xF9  | "00" |  Lock  |  SNI  |  Spill Num (8-bit))  |          Event Num (12-bit)         |

Mass trigger data
|                0x2000                     |   "0000'0000'0"   |   Mass trigger flag (7-bit)  |

HR-TDC block
Sub-hearder
|  0x8000  | "00" |  OerFlow  |  StopDuout  |  Through  |          # of word (11-bit)         |
HR-TDC data
|  Magic word (3-bit)  |  Ch (5-bit)  |                    TDC (24-bit)                      |

LR-TDC block
Sub-hearders
|  0xFC10  | "00" |  OerFlow  |  StopDuout  |  Through  |          # of word (11-bit)         |
|  0xFC20  | "00" |  OerFlow  |  StopDuout  |  Through  |          # of word (11-bit)         |
LR-TDC data
|   0xCC   |  "0"  |   Ch (7-bit)   |   "00000"   |               TDC (11-bit)               |
```

データボディ領域ではsub-headerに続いてその領域のデータが返ってきます。

Mass trigger flagの内訳

ビット番号	フラグ
0	No decision
1	Level2
2	Fast clear
3	Consolation accept
4	Final clear
5	Accept
6	Clear

4.11 Streaming TDC

固有名	0x11dc
メジャーバージョン	0x03
マイナーバージョン	0x05

更新歴

バージョン	リリース日	変更点
v3.5	2021.4.	

動作概要
このファームウェアはtrigger-less DAQ用に開発された外部トリガー無しに連続的に時間測定を行うTDCです。J-PARC MRの遅い取り出しの2秒間の間、1 nsの精度で時間測定を行うことを想定しています。Trigger-less DAQではフロントエンド回路上でトリガーによるイベント選別を行わないため、入力信号全てをデジタイズしPCへデータ転送を続けます。

HULはtrigger-less DAQにおいてdata streamingを行うにはデータリンクスピードが不十分であり、また十分なメモリ搭載していません。このファームウェアは連続時間測定の技術実証のために開発した側面が強いです。今後HULではこのファームウェアの開発は継続せず、**AMANEQ**というtrigger-less DAQ用に開発された回路に引き継ぐ予定です。Trigger-less DAQや連続読み出しに興味のある方は本多へ連絡をお願いいたします。

このような背景があるので、このファームウェアには試験的に実装した機能も存在します。ここでは汎用的に使えそうな部分のみ説明することにします。

ブロック構造
入力はmain-inの上側を利用します。そのほか、テスト入力、VETO入力、スピルゲート入力をNIMINポートから行います。本TDCのチャンネル数は32 chです。テスト入力が有効の場合、main in Uの入力の代わりにNIMIN2からの信号が全チャンネルに配られます。高繰り返しのテスト信号を入れるとデータレートがすぐにリンクスピードを上回るので、利用時にはレートに気を付けてください。VETO入力はHIGHの間入力信号をマスクします。スピルゲートよりもさらに細かく入力信号の選択をしたい場合に利用してください。スピルゲートはJ-PARCの遅い取り出しのスピルゲートを想定しています。スピルゲートがHIGHの状態の時だけ、本TDCはデータを送信します。本TDCは端的にはスピルスタートからの時間を連続的に測るTDCです。そのため、スピルゲートは2秒間ONで2秒間OFFのように周期的にやってくることが前提になっています。

時間計測Online Data Processing (ODP) blockで行われます。ここで入力信号の立ち上がりと立下りの両エッジの時間を測定し、エッジペアを見つけます。この段階でTime-Over-Thresholdが計算され、立下りの時刻情報は破棄されます。以後、データワード内には立ち上がり時刻とTOT値が含まれます。

ODPブロックまでは各チャンネル毎に独立にデータ処理されます。Vital blockではデータパスを1つにまとめ上げ、データ列へハートビートデータという特殊データの挿入を行います。本TDCのコースカウントは125 MHzのクロックで駆動されている16-bitカウンタ (heartbeat generator)によって付与され、500 μs程度で1周します。それ以上の長さで時刻再構成を行うために、heartbeat methodという方式を導入しています。Heartbeat generatorが1周するごとにデータ列にheartbeatデータを挿入します。解析ではheartbeatデータの数を数えることで、スピルスタートからの時間を再構成することが出来ます。Heartbeatデータはデータ列のちょうど区切りの位置に挿入されなければならないため、vital blockにはtime frameの切れ目を見つける工夫が施されています。

データレートがSiTCPのスピードを超えた場合の対処など、さらに詳しい情報は参考文献を参照してください。(R.Honda et al., PTEP, 2021)

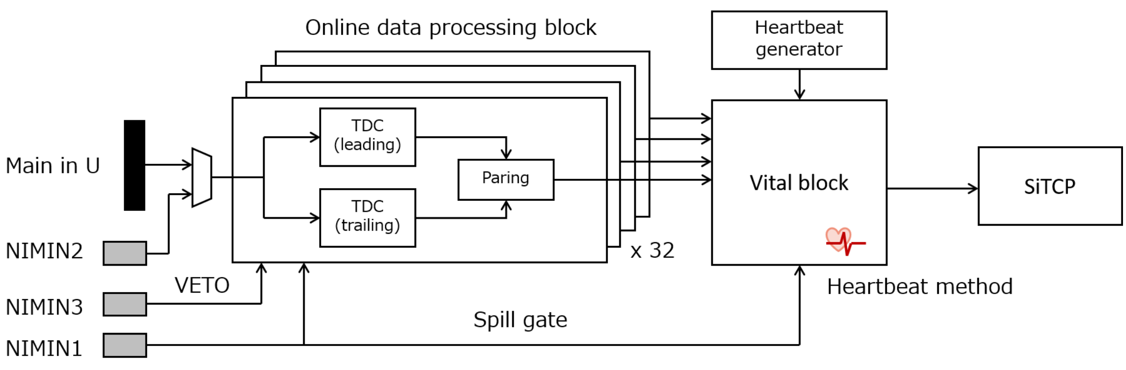

Figure 19: Streaming TDCのブロック図。

Streaming TDC仕様	
TDC精度	0.97... ns
最長スピルゲート長	33秒
最小パルス幅	~4 ns
最大パルス幅	150 ns
ダブルヒット分解能	~7 ns

4.11.1 レジスタマップ

TOT filerとtime-walk correctorはこのUGの中では説明していません。もし利用する場合は参考文献を読んでください。

レジスタ名	アドレス	読み書き	ビット幅	備考
DAQ Controller: DCT (module ID = 0x0)				
SelectTrigger	0x0000'0000	R/W	12	DAQデータ取得を開始するためのゲート。
Online Data Processing block: ODP (module ID = 0x1)				
EnFilter	0x1000'0000	R/W	1	TOT filterの機能を有効にします。
MinTh	0x1010'0000	R/W	8	TOT filterの下限閾値を設定します。この値よりTOTが小さいとフィルターされます。
MaxTh	0x1020'0000	R/W	8	TOT filterの上限閾値を設定します。この値よりTOTが大きいとフィルターされます。
EnZeroThrough	0x1030'0000	R/W	1	TOT値が0だったヒット（立下りエッジが見つからなかったヒット）をフィルターしないようにします。
TwCorr0	0x1040'0000	R/W	8	Time-walk correctorの領域0の補正値を設定します。
TwCorr1	0x1050'0000	R/W	8	Time-walk correctorの領域1の補正値を設定します。
TwCorr2	0x1060'0000	R/W	8	Time-walk correctorの領域2の補正値を設定します。
TwCorr3	0x1070'0000	R/W	8	Time-walk correctorの領域3の補正値を設定します。
TwCorr4	0x1080'0000	R/W	8	Time-walk correctorの領域4の補正値を設定します。

4.11.2 HUL上のスイッチ・LEDの機能

DIP SW2の機能

スイッチ番号	機能	詳細
1	SiTCP force default	ONでSiTCPのデフォルトモードで起動します。電源投入前に設定している必要があります。
2	Enable test in	ONにするとNIMIN2が全チャンネルの入力に接続されます。
3	MCD mode bit-1	MCDメザニンを搭載した際の動作を決めるビットです。通常はOFF (0)に設定します。 • 0b11: Master • 0b10: Repeater • 0b01: Slave • 0b00: Stand alone
4	MCD mode bit-2	MCDメザニンを搭載した際の動作を決めるビットです。通常はOFF (0)に設定します。
5	Enable signal copy	このビットがONだと、下側のメザニンコネクタへ入力信号のコピーが出力されます。別の回路でも信号を使いたい時に利用します。DTLメザニンカードを下側のメザニンスロットへ取り付けてください。
6	Not in Use	
7	Not in Use	
8	Not in Use	

LED点灯の機能

LED番号	備考
LED1	点灯中はTCP接続が張られています。
LED2	機能していません。
LED3	点灯中はスピルゲート入力がHIGHであることを示します。
LED4	機能していません。

NIM-IOへの信号アサイン

Streaming TDCにはIOMが存在しないため、入出力のアサインを変える事はできません。

NIMIO	備考
	NIM-IN
NIN-IN1	スピルゲート入力。
NIM-IN2	テスト入力。
NIM-IN3	VETO入力。
NIM-IN4	未使用
	NIM-OUT
NIM-OUT1	未使用
NIM-OUT2	未使用
NIM-OUT3	NIM-IN1に入力されたスピルゲートのコピー出力。クロック同期を取った後の信号のため立ち上がりタイミングは量子化されている。
NIM-OUT4	未使用

4.11.3 DAQの動作

Streaming TDCはtirgger-less DAQで使用することを想定しているため、データ取得のトリガーという概念は存在しません。 データ転送を行う条件が揃うとFPGA即座にネットワークにデータを送信しようと試みるため、PC側は先にデータ準備が出来ていないといけません。 FPGAがデータを出力する条件は次の3つが全て成立している事です。 1. TCP connectionが成立している。 2. DAQ gateがONになっている。 3. Spill gateがON (NIMIN1の信号がHIGHである)。 本FWを使用する際には、上記順番の通りに処理することをお勧めします。 DAQ gateがONでspill gateがONになるとODPブロックはデータ計測を開始します。 この時点でTCP接続が確立していないと内部バッファにデータが溜まり続けていくため、バッファがあふれてしまう可能性があります。 TCP接続を確立してからRBCPでDAQ gateをONにしてください。

データ構造

Streaming TDCの1ワードは40-bitです。受け取ったデータをそのまま`int32_t`や`int64_t`にキャストできないので、ソフトウェアの記述は工夫してください。 本FWにはspill start, TDC data, heartbeat, busy, spill endの5種類のデータが存在します。 それぞれ先頭の4ビットで識別が可能です。 Heartbeatデータはheartbeat (time) frameの境目に挿入され、spill startから何回目のheartbeatであるかを示すカウンターが下位16-bitに埋め込まれています。 もしvital blockの状態がbusyモードの場合にはheartbeatデータの代わりにbusyデータが返ってきます。

```
Spill start word
MSB                                                                                        LSB
|       0x1      |           RSV (20-bit)          |                0xFFFF                 |

TDC data
|       0xD      | '0' | TOT(8-bit) | Type(2-bit) | Ch(6-bit) |          TDC(19-bit)        |

Heartbeat data
|       0xF      |           RSV (20-bit)          |       heartbeat number (16-bit)       |
Busy data
|       0xE      |           RSV (20-bit)          |       heartbeat number (16-bit)       |

Spill end data
|       0x4      |           RSV (20-bit)          |                0xFFFF                 |
```

TDC dataに含まれるtypeは現状きちんと機能していません。 通常のTDC dataであれば `00` です。そのほかに `01` と `10` のパターンがありますが、これらであった場合そのデータは無視してください。

Busyモード

FPGAが送信しようとするデータ量がデータリンクのスピードを超えると転送が追い付かなくなり、FPGA内部のバッファが溢れます。 このような状況にFWが陥ると、vital blockは通常の動作モードからbusyモードへ状態を遷移させ復帰を試みます。 Streaming TDCではtriggerを止めるためのBUSY信号は存在しませんが、busyモードではbusyデータがheatbeatデータの代わりに現れるようになります。 Busyデータが返ってきたheartbeatフレームではデータ欠落が起きていますが、どのデータをどれだけ落としたかは分かりません。 一部または全てのデータが欠落しているということだけが分かります。 一旦busyモードに移行すると最低3フレーム復帰するために必要とします。 更に詳しい動作については参考文献を参照してください。

データの並び

Streaming TDCから出力されるデータの並びを時間軸上に表した物を図に示します。 Spill gateの立ち上がりを検出するとまずspill start dataが出力されます。 次に最初のheatbeat frame内のTDC dataが続き、frameの境目にheartbeat (busy) dataが挿入されます。 Busy dataが返ってきた場合該当frameではデータ欠落が起きています。

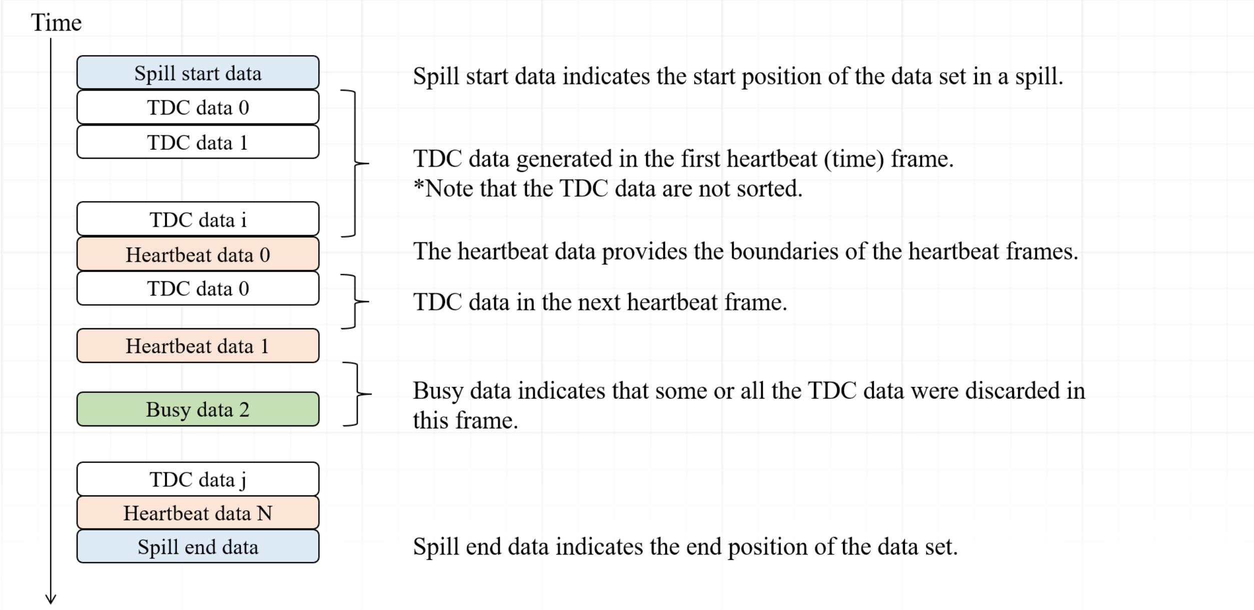

Figure 20: Streaming TDCから出力されるデータの並び順

5. For developers

This chapter provides information for FPGA firmware (FW) developers.

Since HUL uses Kintex7 for its FPGA, it is recommended to develop the firmware using Xilinx Vivado. The installed FPGA, Kintex 7-160T has the size that can be handled Vivado Design Suite WebPACK, without a paid license.

The Xilinx development environments, including Vivado, are released basically with known bugs. Therefore, you should actively update when a new version of Vivado is released, however, when to do so could be an issue. According to Uchida-san (KEK), Vivado is much more stable in version upgrades, compared to the old ISE, and it is probably a good idea to migrate to a new one unless a critical bug is reported. However, I would not recommend to upgrate during a critical development; the synthesis result may change in circuits with critical adjustments. Therefore, I think that the basic policy is to actively update the version of Vivado, only when there is no firmware that has not been confirmed to work.

Vivado consumes a lot of memory during synthesis. It is desirable to have at least 8GB of physical memory, and 16GB or more when running multiple RUNs at the same time. Also, complex firmware takes a long time to synthesize. For example, the HUL MH-TDC consumes 3 GB of physical memory and takes 30 minutes on a ThinkPad X230 (with Core i7-3520M (2.9 GHz)) from logic synthesis to completion of placement and routing. There is a trade off between the mobility and the CPU power/ memory size, which is the only factor to determine the synthesis speed. The best option could depend on the individual cases. Xilinx development tools are primarily intended to use with the CUI, and Vivado can also control the synthesis flow with a scripting language called TCL. Therefore, if you write Makefile etc., it is possible to process jobs in parallel on the Linux server. The firmware size is not comparable to the time of TUL, so it is not recommended to use a cheap laptop.

5.1 Vivado project to be used

item	comment
targe FPGA	xc7k160tfbg676-1
Language	VHDL The language described in the template when the IP is generated. Since Hardware Description Languages (HDL) can be mixed, additional code may be written in other languages, such as VerilogHDL. However, existing modules need to be wrapped in the common language.
Default library	mylib
Synthesis strategy	Vivado synthesize Default

5.1.1 Naming rules

If you intend to upload the FW to hul-official on Gitlab, please align the name with the author's naming rule.

5.1.2 IP generated using SiTCP or IP catalog

SiTCP is the library provided by Bee Beans Technologies, Ltd. Details may be found here.

Since SiTCP is a library provided by a company, it is not included in the FW on gitlab. After cloning, create a directory called SiTCP as shown below, and add the library files into it. The HUL project is set to go to a directory called SiTCP, so if you do this before opening the XPR file in Vivado, you will get no errors. The SiTCP directory is excluded in .gitignore. Please note that FW cannot be merged into hul-official, if its SiTCP library is accidentally included in git.

```
HUL_Skelton.git
├── HUL_Skelton.cache
├── HUL_Skelton.hw
├── HUL_Skelton.ip_user_files
├── HUL_Skelton.runs
├── HUL_Skelton.sim
├── HUL_Skelton.srcs
└── SiTCP
```

A function called **IP core container** has become available from Vivado for IPs generated by IP catalog, but the file size is huge because it is an archive of IP products. Do not use the core container in the HUL FW as it will overwhelm the Git repository.

5.2 Structure of HUL firmware

5.2.1 Top level entity ports

This section describes top level entity port, namely FPGA lead contacts.

Signal name	Signal spec.	Comment
CLKOSC	LVCMOS33	50 MHz clock input from the oscillator on the board.
PROB_B_ON	LVCMOS33	When this signal goes low, the FPGA reconfigures itself from SPI Flash. It is a powerful command equivalent to turning the power on / off. Normally this signal should be high. Note that the configuration will not finish normally, unless the high state is maintained after the power is turned on.
User I/O		
LED	LVCMOS33	It is connected to four LEDs mounted on the top of the front panel.
DIP	LVCMOS33	This is the input signal of the DIP switch. This signal has negative logic and is electrically as shown in the figure below. Remember to set the IOB pull-up as it is required to create the High state.
USER_RST_B	LVCMOS33	It is a pulse input of about 1 ms when SW3 is pressed. This signal is negative logic.
NIMIN	LVCMOS33	NIM input signals of the front panel
NIMOUT	LVCMOS33	NIM output signals of the front panel.
PHY/EEPROM		
The signals categorized in PHY and EEPROM must be kept the same as in the sample project. Also, SiTCP should be implemented in all projects to avoid uninitializing the PHY.		
Differential inputs on board		
MAIN_IN_U	LVCMOS33	Fixed signal input line on the front panel, upper connector.
MAIN_IN_D	LVCMOS33	Fixed signal input line on the front panel, lower connector.
Mezzanine slot		
MZN_SIG_Up/n	differential and single-end signal up to VCCO=1.8V	It is a signal line connected to the upper side of the mezzanine base connector. The signal standard and input / output direction of this signal line are determined by the mezzanine card. Since these signal lines are wired to the HP bank where VCCO is 1.8V. Therefore, differential signals and single-ended signals up to 1.8V are supported; use the LVDS signal standard unless you have a specific need for others. The firmware developer should properly handle the input, output, bidirectional and/or floating or pulled up/down.
MZN_SIG_Dp/n	differential and single-end signal up to VCCO=1.8V	As above.
J0 bus signal		

Signal name	Signal spec.	Comment
J0RS	LVCMOS33	Reading port of J0 bus S1-7. HUL controller uses this line when it is a slave to J0 bus. Trigger and Tag information are received from MTM through this line.
J0DS	LVCMOS33	Writing port to J0 bus S1-7. HUL controller uses this line when it is the master to J0 bus.
J0RC	LVCMOS33	This port reads the C1-2 signal of the J0 bus. J0C line is for BUSY processing only. On the J0 bus, Low is BUSY. Since C1 and 2 should have exactly the same signal, use the OR of C1 and 2 in the module. This line is used when the HUL controller is the master for the J0 bus.
J0DC	LVCMOS33	This port drives the C1-2 signal of the J0 bus. The J0C line is an open collector OR, and low indicates BUSY. Feed the appropriate signal characteristics to C1 and 2. This signal line is used when the HUL controller is a slave to the J0 bus.

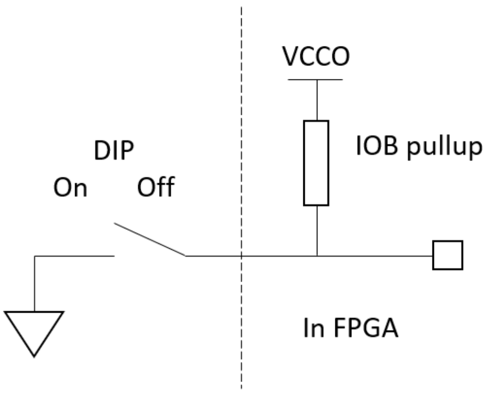

Figure 1: Circuit diagram of DIP SW

5.2.2 SiTCP

SiTCP core requires two types of clocks. One is the system clock, which is synchronized with SiTCP to send and receive TCP and UDP data. The system clock has a lower frequency limit: for 100MHz, minimum 25 MHz and >30 MHz recommended, and for 1 GbE, minimum 125 MHz and >130 MHz recommended. The other is the clock for data communication with the PHY. At 100 Mbps, connect the tx clk sent from the PHY. For GbE, connect a 125 MHz (gtx clk) clock generated inside the FPGA. Gtx clk should be 125 MHz with high precision. The PHY used by the HUL controller allows only +/-100 ppm.

5.2.3 Local bus controller

The author reuses the source code written by Ajimura-san (RCNP) for the Local bus controller (BCT). BCT is the bus interface (Local Bus) managing the local modules inside the FPGA, independent to the raw timing and data lines of the external interface (External Bus). Since External Bus has various timings and data lengths depending on the communication method, BCT absorbs these differences, so that the reusability increases for the source code after the Local Bus. In addition, implement of additional modules requires only a small amount of code changes. The BCT of firmware and FPGAModule class of the software are paired, and are included in the common library of HUL. We encourage you to keep using them in other projects as well.

The BCT bus communication sequence is shown in the figure below. The starting point for BCT to start the bus communication cycle is when the WE or RE of the external bus (SiTCP RBCP) becomes high. The external bus should have the valid addresses and data, so they are stored in BCT. At this time, the information received from the external bus is relocated for the local bus. The relationship between the External bus address and the local bus module ID and local address is as shown in the figure below. From the software side, the RBCP address rearrangement is not seen. BCT uses GetDest to determine from the module ID whether this communication is intended to internally processed by BCT or connect to another module in the FPGA. When BCT is the target, it may change PROB_B_ON status, issue an reset, or get the FW version. If another module is the target, the index of the connection is determined from the module ID. As the next step, SetBus broadcasts the address and data to the local bus, and Connect activates only the WE or RE of the specified index. BCT will wait for a response from the module specified by the index. The connected module starts processing the bus cycle when its WE or RE becomes high. After performing appropriate processing, set ready to high and request the BCT to end the bus cycle. When the BCT receives ready, it enters the termination process and finally returns an acknowledge to the external bus to end one communication cycle.

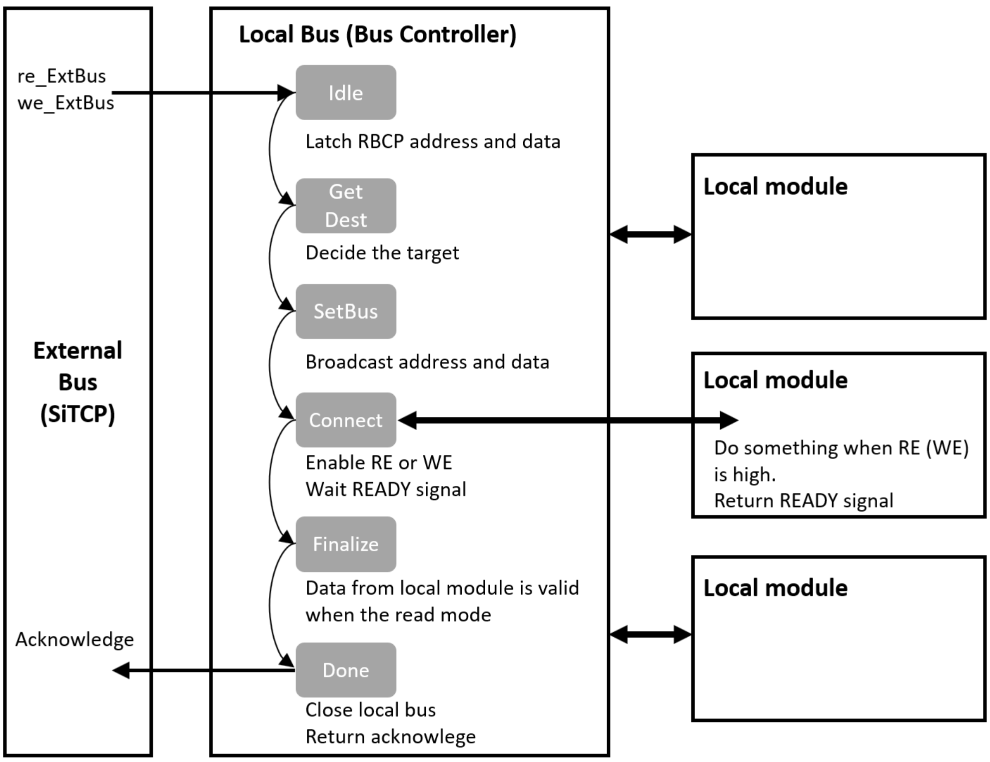

Figure 2: Flowchart of BCT function

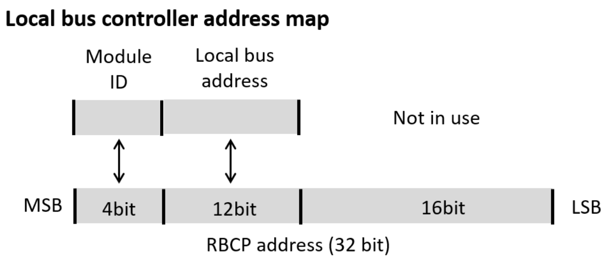

Figure 3: Address map of BCT

Multibyte read (write) on Local bus address

When reading via Local Bus, only 8-bit can be read, so if it is a 4-byte register, it is necessary to perform RBCP communication four times. In HUL, the upper 8 bits of the 12-bit local address are used to identify the register, and the lower 4 bits are used to indicate the number of bytes. For example, if there are 16-bit wide registers A and B in a local module with a module index of 0x8, the BCT and RBCP addresses will be:

BCT address	RBCP address	comment
0x000	0x8000'0000	Register A Byte-1 (Lower 8-bits)
0x001	0x8001'0000	Register A Byte-2 (Higher 8-bits)
0x010	0x8010'0000	Register B Byte-1 (Lower 8-bits)
0x011	0x8011'0000	Register B Byte-2 (Higher 8-bits)

An example of HDL of a local module when reading multiple bytes.

```
when kReadLength(kNonMultiByte'range) =>
  if( addrLocalBus(kMultiByte'range) = k1stbyte) then
    reg_length_read(7 downto 0) <= dataLocalBusIn;
  elsif( addrLocalBus(kMultiByte'range) = k2ndbyte) then
    reg_length_read(kWidthRead-1 downto 8) <= dataLocalBusIn(kWidthRead-1-8 downto 0);
  else
end if;
```

The `FPGAModule` class included in the `hul_software` packages supports multi-byte read / write, and the lower 4-bits of the BCT address are automatically incremented.

5.2.4 Adding a module to Local Bus

Change the following if a module is being added:

Variables in defBus.vhd

item	comment
kNumModules	Number of modules except BCT itself
Module ID	ID starting with "kMid" This parameter turns to the RBCP address. There is no need that the address is continuous.
Leaf	ID to specify the index on Local bus The number has to be continuous.

BusController.vhd

Add a when structure for the additional module, in the else structure of GetDest state of the state-machine.

toplevel.vhd

Module body is conected to LocalBus. Index of the array specifies the new Leaf ID.

6. Software

This chapter describes Linux software for control.

Environment for Development

- CentOS7
- gcc version 4.8.5

The source code for HR-TDC depends on CERN ROOT, because it creates a ROOT file. In an environment where ROOT cannot be installed, exclude HR-TDC from the make target.

Direcotry structure

"CoreLib" provides the common library used by this software package. "Common" provides programs to control common elements in each firmware, such as SDS and FMP control. In addition, the source code dedicated to each firmware exists in the directory with the FW name.

```
hul_software.git
├── Makefile
├── CoreLib
├── Common
└── FW directories
```

6.1 Source files

The source files may be categolized as

- library-like files that should be copied as they are when porting to other data acquision (DAQ) systems,
- files that can be written in any way as long as they work, and
- files that are for debugging and do not need to be ported.

Only the first category will be explained.
All executables are designed to display usage when started with no arguments.

CoreLib

rbcp.hh

The structure of the RBCP packet. Do not change.

Uncopyable.hh

A method that prohibits copying constructors and assignment operators in user-defined classes. This implementation is old, and the standard features of c ++ nowadays may do the same; this will be eliminated in the future.

BitDump

This class returns a bit string to the standard output when an integer value is given. It was originally used in Unpacker, and used in UDP RBCP. Useful for debugging.

Utility

Shows a progress bar or blinks text on the console. Used in the Flash Memory Programmer.

UDPRBCP

It is the class for UDP communication of SiTCP and is an important source code. The constructor requires an IP address, upd port, and the last argument is display mode, defined in `UDPRBCP.hh` as an enumeration called `RbcpDebugMode`.

- `NoDisp` displays nothing during UDP communication.
- `Interactive` displays some information about the operation.
- ` Debug` displays all information available.

For usual use, `NoDisp` will be fine. UDPRBCP provides the low-level UDP communication in SiTCP. HUL may be controlled with this class, but primitively.

FPGAModule

This class provides BusController (BCT) level communication function. The constructor takes a UDPRBCP as an argument to include UDPRBCP. Its main function is to support multi-byte reading and writing through BCT.

```
int32_t WriteModule(const uint32_t local_address,
                    const uint32_t write_data,
                    const int32_t  n_cycle = 1);
```

A method to write a register. The first argument is the RBCP address. See the register map in Chapter 4 and `RegisterMap.hh` described below. The second argument is the register value to write. The third argument is the number of bytes be written. The return value is the result of `UDCPRBCP::DoRBCP`. This function is overloaded for 32-bit and 64-bit registers. The range in which `n_cycle` can be taken is 1-4 (up to 4 bytes=32 bits) and 1-8 (up to 8 bytes = 64 bits), respectively.

```
uint32_t ReadModule(const uint32_t local_address,
                    const int32_t  n_cycle = 1);
```

A method to read a register. The first argument is the RBCP address. The second argument sets the number of multibyte to read. The read result appears in the return value of the function. Independent to n_cycle value, the return is in the type of `uint32_t`. This function is also overloaded for 32-bit and 64-bit.

```
int32_t WriteModule_nByte(const uint32_t  local_address,
                          const uint32_t* write_data,
                          const int32_t   n_byte);
```

A function for writing multiple bytes to the same address. This function does not increment the address on every write. The assumed write destination is FIFO.

```
int32_t ReadModule_nByte(const uint32_t  local_address,
                         const int32_t   n_byte);
```

A function for reading multiple bytes from the same address. This function does not increment the address on every read. The assumed read source is FIFO. The read data is stored in a class variable called `rd_data_`. Get the iterator with `GetDataIterator` to access the data.

SiTCPController

A function to directly access the reserved area of SiTCP. EraseEEPROM is a function to erase all EEPROM. Functions for reading and writing to the EEPROM area are not developed because they are dangerous. Using the SiTCP Utility (Chapter 3) is recommended.

```
void ResetSiTCP(const std::string ip)
```

A function to soft reset SiTCP.

```
void WriteSiTCP(const std::string ip,
                const uint32_t addr_ofs,
                const uint32_t reg)
```

A function to write a 1-byte register to the SiTCP reserved area. The address starts from 0xffffff00, and the offset from there is specified by `addr_ofs`. The EEPROM area cannot be accessed by this function.

```
void ReadSiTCP(const std::string ip,
               const uint32_t addr_ofs)
```

A function to read a 1-byte register from the SiTCP reserved area. `addr_ofs` has the same meaning with WriteSiTCP.

```
void EraseEEPROM(const std::string ip)
```

A special function for erase all EEPROMs. The SiTCP license file will need to be rewritten, because IP and MAC addresses and license information will be erased. This function is used to solve the problem reported in the SiTCP community of BBT, titled: "Once a TCP connection is established, it cannot be reconnected for a while". See the relevant thread on the BBT web page for details.

Common

The controlling finctions for FlashMemoryProgrammer (FMP) and Self Diagnosis System (SDS) are collected. Since these are common parts of HUL's FW, the registers and register address are stored in `RegisterMapCommon.hh`. In the following, behavior of major functions are explained, as they provide examples of control on HUL.

assert_bctreset

Asserts soft reset signal `BCT::Reset` in the FW.

erase_eeprom

An executable that calls the EraseEEPROM of SiTCPController.

reconfig_fpga

An executable that actively reconfigures the FPGA from SPI flash memory, as usually happens when the power is turned on. This program is used when the FPGA received a radiation damage, or a new MCS is downloaded to the SPI flash memory using FMP, etc.

read_xadc

Access the XADC interface in the SDS to get the FPGA temperature, VCCINT voltage, and VCCAUX voltage.

read_sem

Access the SEM interface in the SDS to read the number of single event upset (SEU) corrections, the SEM's status (watchdog), and whether uncorrectable radiation damage has been detected (uncorrectable). The number of SEU correction returns to 0 by issuing reset signals higher than or equal to the BCT reset or accessing for write to `SemRstCorCount`. If "Uncorrectable" flag is set to 1, the FPGA needs to be reconfigured.

reset_sem

Assert a soft reset to SEM.

inject_sem_error

Use SEM to artificially generate a SEU for debugging. Since DRP is used to access SEM, more could be done with SEM. See Xilinx PG036 for more information.

flash_memory_programmer

A program to writie a MCS file to SPI flash memory via SiTCP. It converts the MCS (text file) to binary, check the SPI memory model number, memory erase, program, readback and verify in order.

mcs_converter

A program to convert a MCS file to a binary in advance.

check_spi_device

Check the model number of the SPI flash memory on HUL. Use it when you don't know what's on board

verify_mcs

Read back from SPI flash memory and check for consistency with the specified MCS file.

6.2 Programs for each FW

Most FWs only implement the `debug` and `daq` executables. `daq` is an executable to acquire data. It is a sample code of data acquision (DAQ). `RegisterMap.hh` contains the FW-specific register address and registers. This section describes FW-specific programs.

DaqFuncs.cc

This is the collection of source codes for DAQ. It is probably better to use the ConnectSocket, Event_Cycle, and receive functions as they are. Also, SetTdcWindow in MH-TDC should be as it is. Basically, the flow one RUN is, configuring the measurement block, open the gate with DCT, keep calling EventCycle, and close the gate with DCT when acumulation is over.

Since the buffer in HUL is flushed by calling EventCycle until it times out after closing the Gate, be sure to perform `while (-1! = EvenCycle)` processing. Otherwise, events remained in the previous RUN may be returned at the beginning of the next RUN.

daq

Read data from the HUL. Give the IP address, RUN number, and number of events as argumentsa as follows `./bin/daq [ip address] [RUN no] [# of events]`. The output file will be created as `data/run1.dat` in the `data` sub-directory of where the command was issued. So, the programs is expected to be run under each firmware directory, where the `data` sub-directory has been created by `make` command.

During data aquisition, read data are displayed once in some events for debuging.

6.2.1 HR-TDC

initialize

Initialize the DDR receiver and calibrate the LUT for the estimator using the calibration clock. Must be run once after the power is up.

decoder

A decoder provided for debugging. Give the RUN number as follows `./bin/decoder [RUN no]`, the program reads a data from the `data` directory and reate a rootfile into the `rootfile` directory. The programs is expected to be run under each firmware directory.

The structure of TTree Branches

Branch name	Description
through	Indicates register setting for `TDC::Through` in mezzanine HR-TDC. Index 0 and 1 correspond to slot U and slot D, respectively.
stop_out	Indicates register setting for `TDC::StopDout` in mezzanine HR-TDC. Index 0 and 1 correspond to slot U and slot D, respectively.
over_flow	Indicates over flow status stored in sub-headers. Index 0 and 1 correspond to slot U and slot D, respectively.
Decoded common stop TDC data.	
stop_fine_count	Fine count (lower 13-bit of TDC data) for common stop data. It's avairable when TDC::StopDout is true.
stop_estimator	Estimator value for common stop data. It's avairable when TDC::StopDout is true.
stop_coarse_count	Coarse count (upper 11-bit of TDC data) for common stop data. It's avairable when TDC::StopDout is true.
common_stop	TDC value for common stop. It's avairable when TDC::StopDout is true.
Decoded data for leading edge measurement.	
num_hit	The number of data for leading the edge measurement.
num_hit_ch	The number of data in each channel for leading the edge measurement.
channel	Channel number for the decoded data word. The index number indicates the order in which they were decoded.
fine_count	Fine count (lower 13-bit of TDC data) for the TDC data word. It's avairable when TDC::Through is true. The index number indicates the order in which they were decoded.
estimator	Estimator value for each TDC data word. It's avairable when TDC::Through is false. The index number indicates the order in which they were decoded.
coarse_count	Coarse count (upper 11-bit of TDC data) value for each TDC data word. The index number indicates the order in which they were decoded.
tdc_leading	TDC value for each TDC data word. The index number indicates the order in which they were decoded.
Decoded data for trailing edge measurement.	
tdc_trailing	TDC value for each TDC data word. The index number indicates the order in which they were decoded.
Branches starting with 't' are decoded data values for the trailing edge measurement.	

7. Practical Usage

In this chapter, a few tips are introduced, which are good to know when using HUL.

7.1 Generation and download of MCS with Vivado

Generation of MCS

How to generate MCS using Vivado is explained. An alternative tool, iMpact, is omitted, because few people use it.

Open the project in Vivado and select tool → Generate Memory Configuration File. Fill information as in Figure and press OK to generate MCS. This screen shot is taken from Vivado 2020.1.

As of 2021, there are three kinds of SPI flash memory used in HUL. Each should be configured in Vivado as follows.

IC on board	Manufacturer	Part ID on Vivado
N25Q128A	Micron	mt25ql128-spi-x1_x2_x4
MT25QL512	Micron	mt25ql512-spi-x1_x2_x4
S25FL256S	Spansion	s25fl256sxxxxxx0-spi-x1_x2_x4

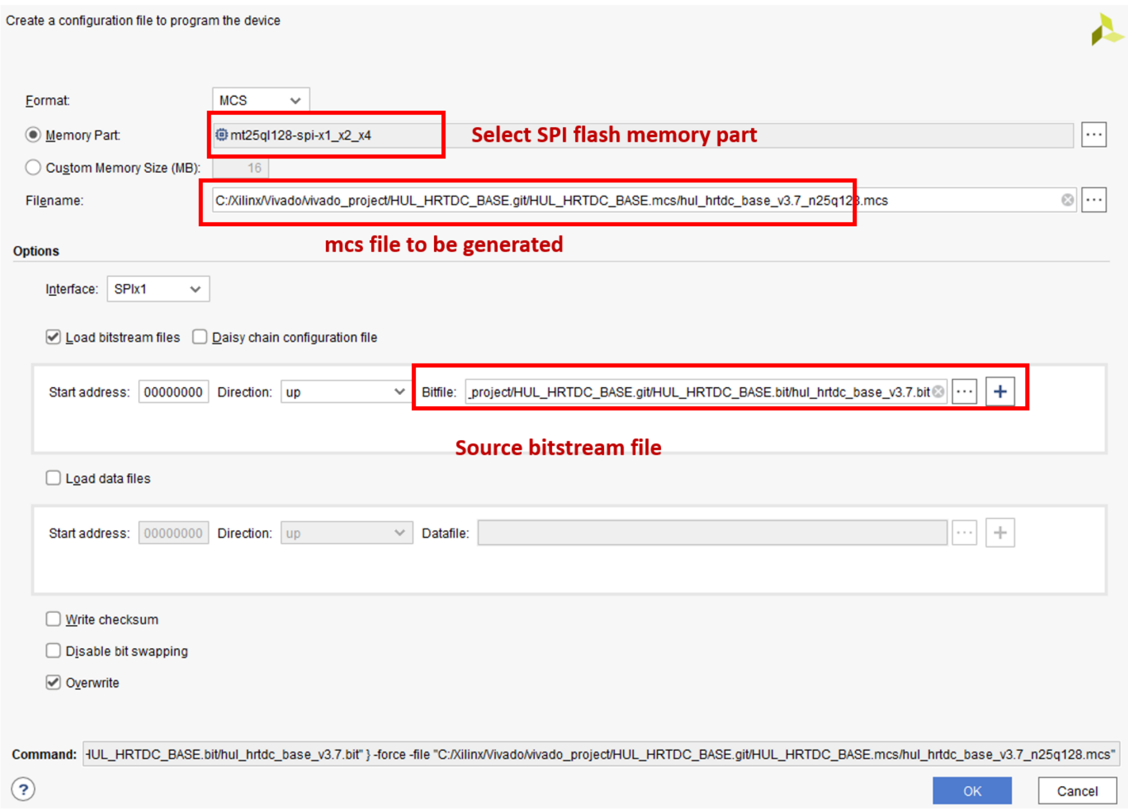

Figure 1: Generation of MCS

How to download MCS by Hardware manager

Connect the JTAG cable with the HUL powered on. The LED of the download cable will turn green, if correctly connected. If it stays orange, the USB connection may be taken by the client OS, if virtual machine is used on PC. With the green LED on, access the FPGA with "Open Hardware manager" → "Open target" in Vivado. Kintex7 (XC7K160T-1) should be discovered. If you want to write a Bit stream file, right-click the FPGA, assign the Bit stream file, and write it. MCS file should be downloaded to the SPI flash memory, however, the memory will not appear on the screen at this point. This is because the SPI flash memory does not exist yet on the JTAG chain.

To make SPI visible, add the configuration memory device. Right-click on the FPGA and select "Add configuration memory device". A screen like Figure will appear, select the appropriate device in the Memory Device (MCS has to be created accordingly). Select both MCS and PRM files, and leave everything else as default. Then OK to write; it takes about 10 minutes to write.

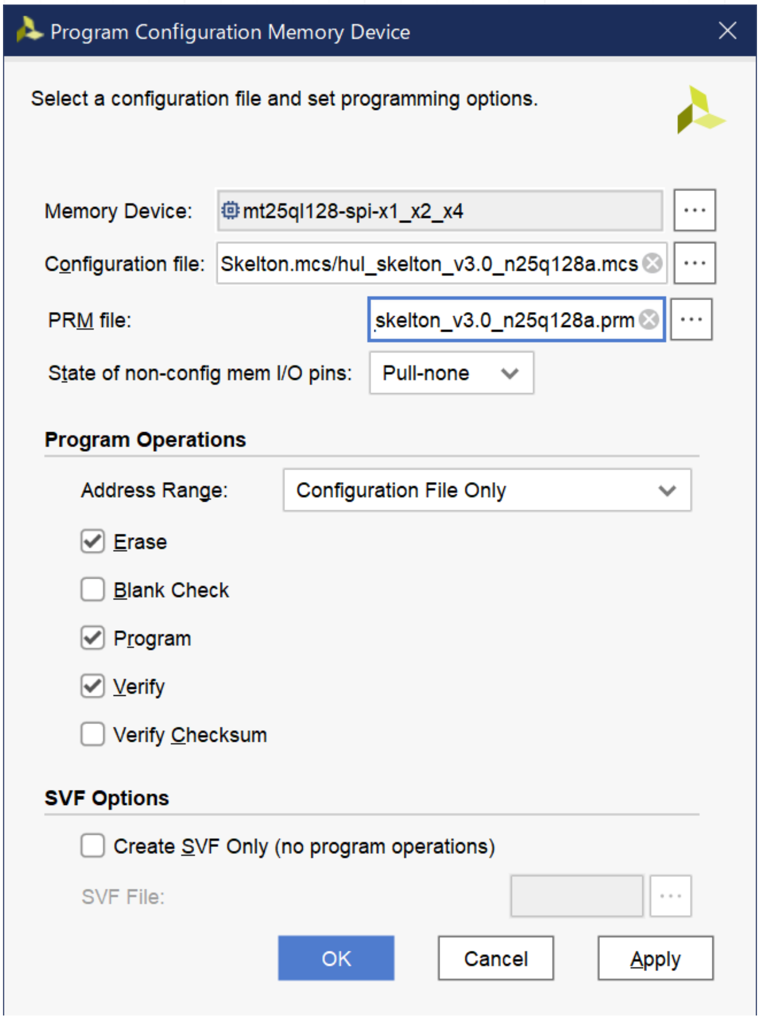

Figure 2: Configuration Memory Device

For N25Q128A

Micron's N25Q128A is the chip selected when designing the HUL, but it has been discontinued. The latest version of Vivado does not support this chip anymore. However, N25 series is internally compatible with the MT25 series, which is the successor chip. When generating the MCS for N25, select **mt25ql128** and make it as MT25QL. Also, when downloading use the setting for MT25QL.

For MT25QL512

In late 2017, the mass-purchased board changed the SPI flash memory to MT25QL. When generating MCS file, Figure select **mt25ql512_spi-x1_x2_x4** for this SPI memory. In Hardware manager, select the same for memory devise in Figure.

For S25FL256SAGNFI001

Since 2018, SPI flash memory chip is changed to Spansion S25FL256SAGNFI0. Select **s25fl256sxxxxxxx1-spi-x1_x2_x4** when making MCS and downloading it by Hardware manager.

Failure of FPGA initialization when JTAG is connected, Hardware manager is up and HUL is powered on

You will encounter this problem if you turn on the power with the download cable connected while the Hardware manager is running and the JTAG server is open. Even if the Hardware manager is running, it can be avoided if the server is closed.

7.1.1 MCS download to Mezzanine HR-TDC

As of 2021, the mezzanine HR-TDC has two types of flash memory, as described in Chapter 2. Please select the following part number when generating MCS and downloading.

IC on the board	Manufacturer	Part ID on Vivado
N25Q128A	Micron	mt25qu128-spi-x1_x2_x4
MT25QU256A	Micron	mt25qu256-spi-x1_x2_x4

7.2 MCS download via SiCPT by FMP

MCS may be downloaded to SPI flash memory via the network (RBCP) by using the Flash Memory Programmer (FMP). Without this technology, it was necessary to insert the USB download cable into the JTAG socket on the board, so someone must go and reach to the board. Replacing the FW during the experiment was a big task. By using FMP, you can download MCS from the measurement room remotely, and reconfigure the FPGA without touching the hardware.

Prepare the same MCS file as writing with JTAG. The executable to write to SPI flash memory is in `hul_software.git/Common/bin`. Execute as follows, with a -b option if a binary file has been generated by `mcs_converter`.

```
./bin/flash_memory_progorammer [ip-address] [mcs file path]
```

If the download is successful and the verify result matches the original MCS, the following is displayed. If communication is interrupted or a mismatch happens, please retry again until being successful. At this stage, the firmware is only downloaded to the SPI flash memory, and not reflected in the FPGA. After successfully downloaded, use `reconfig_fpga` to reconfigure the FPGA.

```
[tohoku@centos7 Common]$ ./bin/flash_memory_programmer 192.168.10.16 mcs/hul_mhtdc_v3.4_n25q128.mcs
#D: [FlashMemoryProgrammer::ReadMCSFile()
 MCS file:    mcs/hul_mhtdc_v3.4_n25q128.mcs
 Binary size: 6.69257MB (6692572 bytes)

#D: [FlashMemoryProgrammer::CheckSpiDevice()
 Target device:  n25q128a

#D: [FlashMemoryProgrammer::EraseFlashMemory()

#D: [FlashMemoryProgrammer::EraseFlashMemory()
 Flash memory erased.

#D: [FlashMemoryProgrammer::ProgramFlashMemory()
 Proceeding program.
[########################################]100%
#D: [FlashMemoryProgrammer::ProgramFlashMemory()
 Program done.

#D: [FlashMemoryProgrammer::VerifyMCS()
 Proceeding verify.
[########################################]100%
#D: [FlashMemoryProgrammer::VerifyMCS()
 Readback done.

#D: [FlashMemoryProgrammer::VerifyMCS()
 MCS data is Verified.
```

Figure 3: Console display after successful MCS download with FMP.

Known issues

It takes a certain amount of time to write a page in the flash memory, and it may not be completed before UDP RBCP turns one cycle, depending on the machine specifications of the PC and the communication environment. The current FMP is not designed to block the next write operation while writing a page, and if the write request for the next page is issued too early, communication may be interrupted or a UDP RBCP bus error may occur. Currently, `usleep` is forcibly delaying the next write, but it is not a good solution and it takes a long time to write. In the future, the issue may be handled on the FPGA side, but not yet tackled.

7.3 A few How-to's in usage

A simple test

Here is a simple test method for HUL RM, HUL Scaler, and HUL MH-TDC. These firmwares will work without inserting the module into the VME crate. In `daq_func.cc`, select `TRM::L1Ext` to be the only input in `TRM::SelectTrig`. Connect the trigger signal to NIMIN1. To monitor the data, execute `./bin/daq`.

Having multiple HUL modules in one crate

With multiple HULs inserted in the crate, only one can be the J0 bus master, and others to be in the slave mode. If there are two masters, J0 bus will be short-circuited. (If KEK VME crate is not used, there will be no issue.)

How to set Master mode

Mount HRM on Slot-U.

- Turn on all DIP SW1. (If this is ON, it will be in driver mode for J0 bus.)
- Turn on the 2nd bit (mezzanine HRM) and 4th bit (Bus BUSY) of DIP SW2. (The firmware must support HRM.)
- Set `TRM::EnRM` to be 1 and EnJ0 to be 0.

How to set Slave mode

- Turn off all DIP SW1.
- `TRM::EnJ0` to be 1 and EnRM to be 0. Try force busy (DIP SW2 bit 3) to be ON on a Slave module, and check HRM busy LED to be ON (correctly configured).

A problem after module reset, when receiving Level2 trigger in KEK VME J0 bus.

There was a problem of DAQ not run after a module reset (e.g. BCT reset) was issued. The version 3 firmware has solved the problem. Please report if there is a problem.

Event slips

It is possible to monitor event slips via J0 bus and HRM Tag. If the event numbers do not agree between the modules, an event slip has occured. Since HUL firmware has a multi-event buffer, RUN Start/Stop will not solve the slip. Clear the internal buffers by `BCT::Reset`.

How to use HR-TDC

HR-TDC will hang up easily in mis-operations, because the system has two FPGAs and there is a initialization procedure. For the first-time use, the procedure written here is recommended. **Be sure to use a VME crate.**

1. Insert the Mezzanine HR-TDC into the HUL and download the MCS to each FPGA. It is recommended to start with MCS as the BCT may hang if you use the bit stream after turning on the power. On Mezzanine HR-TDC, specify mt25qu128-spi-x1_x2_x4, when downloading MCS.
2. Turn the crate power on / off once.
3. Prepare the software. If two Mezzanine HR-TDCs are installed, you can use the distributed C++ code as it is. If only one is installed, set `EnSlotUp / EnSlotDown` in RegisterMap.hh to be false for the empty slot.
4. Run `./bin/debug`. The version of the HUL and mezzanine firmwares will be displayed. If a bus error is issued here, there is a configure problem. First check the software, and if there is no mistake, try detach / install the mezzanine card(s). The high speed communications through the mezzanine slot tend to cause the bus error, and require a good contact.
5. Execute `./bin/initialize` to initialize DDR and generate estimator using calibration clock. Make sure both are successful.
6. The distributed C++ code assumes NIMIN1 should have a Common stop. Connect the trigger to NIMIN 1 and execute `./bin/daq [run no] [# of events]`. This should return the data in the subdirectory, such as ./data/run[run no].dat.
7. To create a ROOT file from the binary data, execute `./bin/decode [run no]`. It will create a ROOT file under `./rootfile/run[run no].root. See the software section for more detail about the TTree structure.

7.4 Miscellaneous

How to purchase modules from GND Ltd

Send an email to GND Co., Ltd., telling the **GND catalog number** and **its number of purchase**. If in stock, the modules will be delivered immediately. HUL controller module includes the SiTCP license fee in the price. If the module is out of stock, we must consult the manufacture plan. Independent orderwill result to a high price, because the initial cost for production will be split to the number of modules manufactured.