Development of Event Receiver on Zynq7000 Evaluation Board

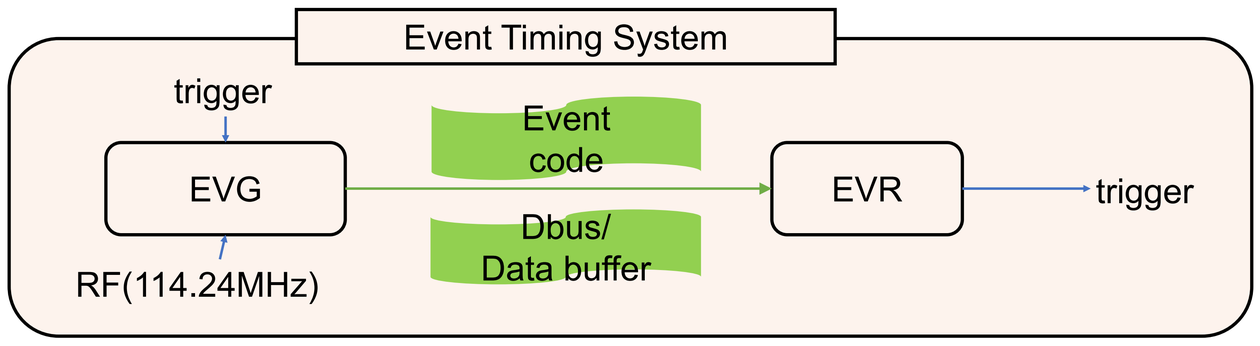
Hitoshi Sugimura (KEK)

Motivation of Development

1. Timing system in SuperKEKB project is using Event Generator(EVG230) and Event Receiver(EVR-230RF) developed by Micro Research Finland Oy. This project is on going, and operating while changing the specifications.
2. The example of change specification is fine delay tuning less than 10 ps, and take unused bit information like distributed bus bit.
3. I want to respond quickly and relatively cheaply by developing self made modules using FPGA
   - If we have knowledge of FPGA language (VHDL or verilogHDL), we can do the job of subcontracting by ourself.
4. In the future, the event system will be integrated with devices such as BPM and RF.

Requirement for Event Receiver

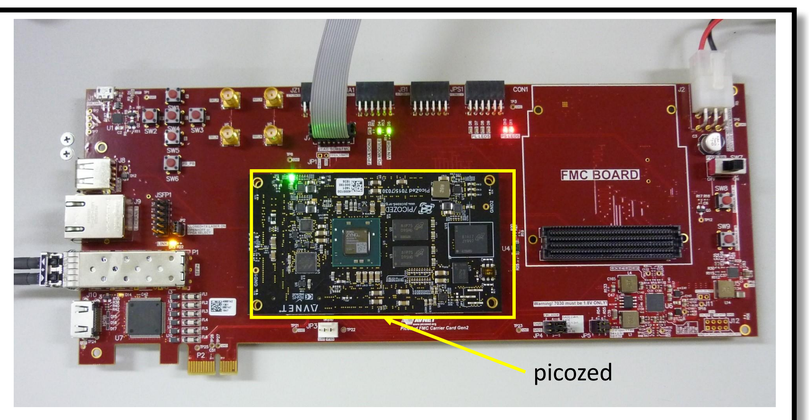
Since the event code is transmitted in synchronization with the RF clock of 114.24 MHz, including 8b10b conversion, a communication speed of 2.5 Gb/s is required. Low jitter of less than 20 ps is also required



Open Source Event Receiver

MRF released the source code of the FPGA including GTX configuration so that the MRF module's user could make event receivers by themselves. The released code uses Zynq7000 to describe the circuit. GTX is included in Kintex-7 and Zynq7000 also have a part of equivalent to Kintex-7 in programmable logic part (only for Z7030 and Z7040 series).

The merit of Zynq is that it can be controlled by the ARM core. This can be standalone module without going through the bus control, and also can be run in EPICS IOC in Zynq. Since the released code was developed using AVNET's picozed, we decided to purchase a similar board and proceed with development based on open source code.



The carrier card model number is AES-PZCC-FMC-V2-G, And picozed model number is AES-Z7PZ-7Z030-SOM-I-G

Configuration of Event Stream

Event stream is sent continuously as one cycle of 8-bit Event Slot and 8-bit Distributed bus or Data buffer. Distributed bus and data buffer are sent alternately.

Example of data stream transmitted from Event Generator

K28.5 is sent every 4 cycles from the continuously sent data string, thereby clearly indicating the break of the data.

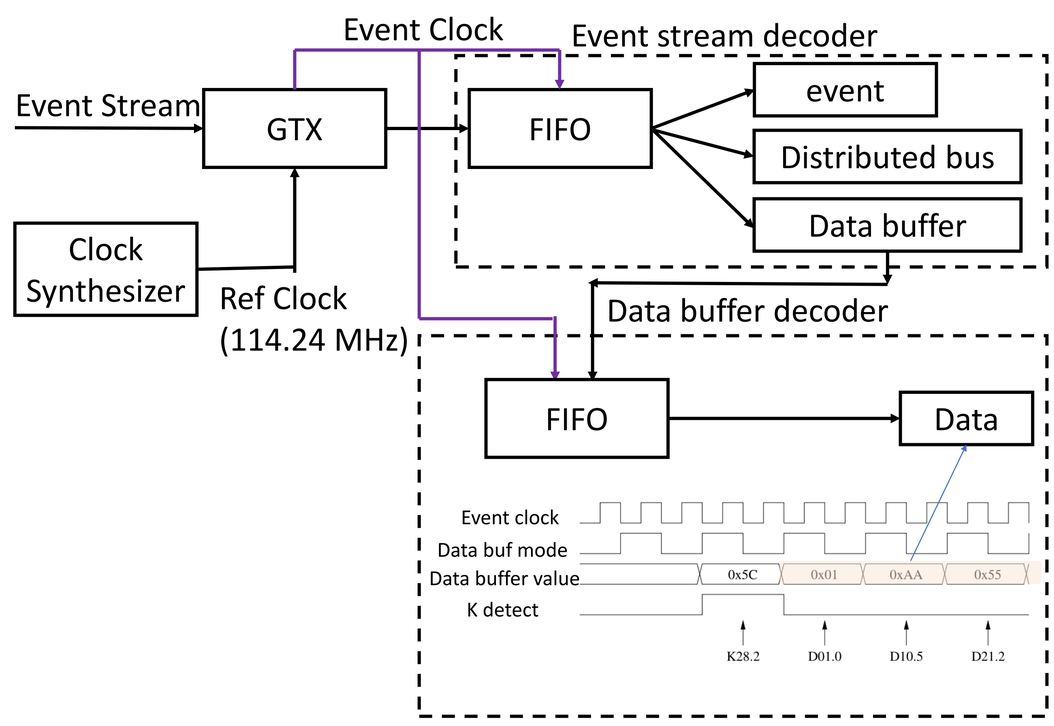
Event code "16" is transmitted. When there is no event code, "0" (D00.0) is transmitted.

Cycle	Event Slot	Distributed Bus / Data Slot
0	K28.5 Sync character	D00.0 Distributed bus byte
1	D00.0 Null event	D00.0 Data buffer, null data
2	D00.0 Null event	D01.0 Distributed bus byte
3	D00.0 Null event	K28.2 Start of Data buffer
4	K28.5 Sync character	D01.0 Distributed bus byte
5	D16.0 Event code 0x10	D00.0 Data buffer, null data
6	D00.0 Null event	D01.0 Distributed bus byte
7	D00.0 Null event	K28.1 End of Data buffer
8	K28.5 Sync character	D01.0 Distributed bus byte

time

Data buffer can send up to 2 kByte, D00.0 is sent when not in use.
K28.2 is sent at the beginning of the data buffer and K28.1 is sent at the end.

GTX setup

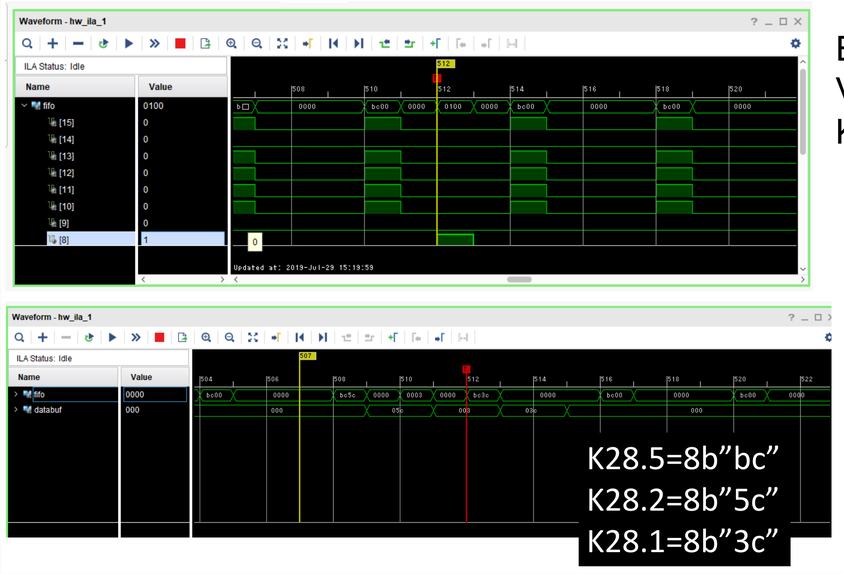


At first, comma alignment is performed using K28.5 sent from EVG as a synchronization event every 4 cycles. If K28.5 appeared, the data is distributed to event code and data buffer and distributed bus bit in each 16-bit of data frame. The event clock is generated according to reference clock. The input reference clock rate was set to 114.24 MHz using the universal frequency converter (IDT-8T49N242) mounted on the carrier card.

Since the data buffer has a maximum size of 2 Kbytes, reserve that amount in BRAM in advance.

Monitoring by Logic Analyzer

Since there is no digital output in picozed and carrier card, output signals cannot be seen. Therefore, by setting up ILA(Integrated Logic Analyzer) provided as an IP core tool, I try to monitor a data buffer and event code reception from EVG on the test bench.



Event signal monitored using Vivado, and 0x3 after receiving K28.2 is data buffer value.

K28.1 is received subsequently. The lower is obtained waveform by extracting the databuffer from the event signal

We are able to confirm that the reception and distribution of the data buffer were successful.

Future prospects

- We have created a mechanism that takes in Event Stream using GTX and distributes data to event signal, data buffer and distributed bus bit.
- The event signal is correctly received using the logic analyzer.
- In the future, we will develop a mechanism to extract received event codes as signals.
- Test whether FMC card with Digital IO can output signal.
- We want to evaluate timing jitter and make it as an EVR standalone modules.