Embedded Event Receiver (EEVR)

evr320
Revision 2.4

Firmware Data Sheet

PSI, 03.02.2021

Content

Table of Contents

1 Introduction

The SwissFEL accelerator placed at PSI use a timing system provided by Micro-Research Finland [1]. The timing system consists of one Event Generator (EVG) and several Event-Receivers (EVR). Both are realized as a VME card. The communication between timing components is realized with fiber optical links. Some of the accelerator systems which need the timing system are realized as a stand-alone device with no VME bus and no access to the VME EVR. Most of these devices have optical connector which can be used to connect to the timing system. Therefore an Embedded Event Receiver (EEVR) was implemented which offers subset of EVR functions. The EEVR is a VHDL component which can be integrated with existing Xilinx FPGA projects.

1.1 Features

EEVR has the following properties:

- Distributed Bus decoder with update rate at 142.8 MHz
- Decoder of four user defined events with events table configurable in run-time
- Local memory for segmented data buffer
- Ready for 142.8 MHz clock recovery with deterministic phase
- Optional Features:
  - Event Recorder
  - AXI4.0 interface
  - TOSCA-II Interface to use with IFC1210
- Portable to Xilinx FPGA:
  - Kintex
  - Virtex-6

1.2 Definitions, acronyms, and abbreviations

FPGA	Field Programmable Gate Array
EVG	Event Generator
EVR	Event Receiver
EEVR	Firmware Event Receiver
AXI	Advanced eXtensible Interface

1.3 References

[1] “Event System with Delay Compensation– VME-EVM-300, VME-EVR-300, mTCA-EVR-300, PCIe-EVR-300DC, Technical Reference VME-EVM-300 Firmware 22030207, VME-EVR-300 Firmware 12070207, PCIe-EVR-300DC Firmware 17060207, mTCA-EVR-300 Firmware 18070207”, Micro-Research Finland, 3. May 2017

1.4 History

Revision	Date	Author	Description
2.0	26.05.2017	G. Marinkovic	Version tested with GPAC board, Event Generator and IOXOS IOC.
2.1	03.04.2018	P. Bucher	Event Recorder Functionality added, tested on ifc1210 with Event Generator.
2.2	30.10.2019	J. Purtschert	Latency Measurement register added
2.3	25.11.2019	B. Stef	Generics for event pulse length and delay added on ifc1210 wrapper
2.4.	03.02.2021	J. Purtschert	Correct Latency counter description

2 Functional Description

The EEVR was implemented in VHDL. It contains device specific primitives such as MGTs and FIFOs. Therefore its portability is limited - see feature list in section 1.1. The component is intended to be instantiated in other bigger projects which need timing information.
The Event Recorder functionality can be added on instantiation to verify the appeared events and their arrival time (in clock cycles) form the configurable Start-of-Sequence (SOS) Event. This monitoring functionality applies as well for the arrival time for the Segments of the Segmented Data Buffer.
As an option the EEVR component is wrapped in a Xilinx Vivado IP component with slave AXI4 interface or can be used with the TOSCA-II Interface to use with IFC1210 based Projects.

2.1 Firmware Description

The firmware consists of MGT component, decoders and AXI interface.

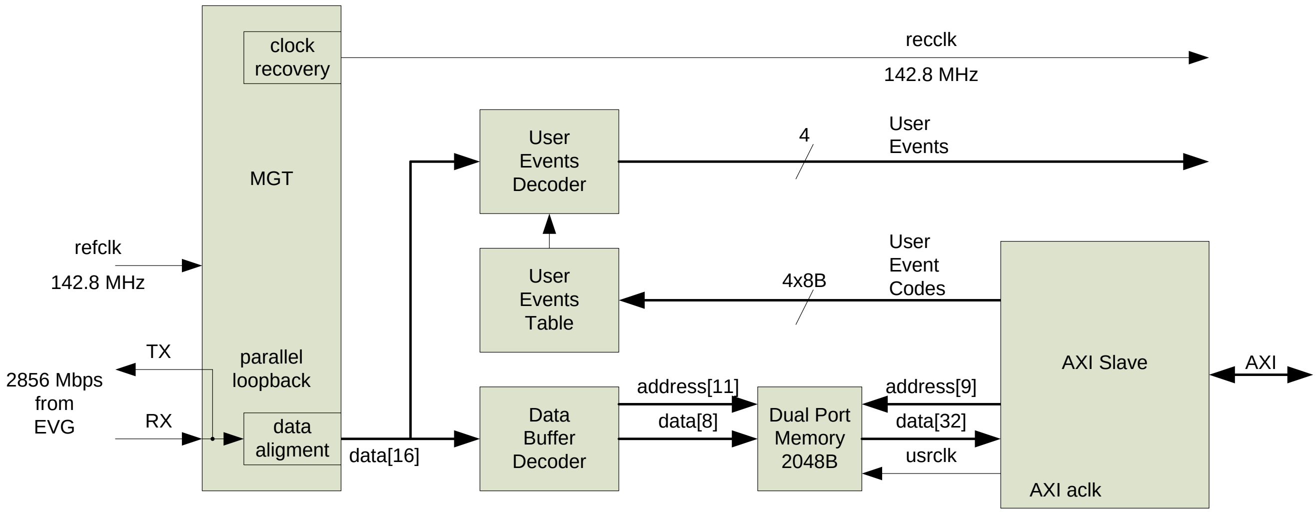

Figure 1: Block diagram of the evr320 firmware

The MGT component is built of MGT primitive and associated logic. The MGT primitive instantiation is specific for each FPGA chip. Currently only one type of the MGT primitive is supported: GTX for Kintex-7 FPGAs. The MGT component does not use internal RX elastic buffer and the comma alignment is done in user logic. The reason for that is to get the recovered clock 142.8 MHz with deterministic phase with respect to the source. All signals derived from the serial data stream are also phase align to the recovered clock. The data buffer memory was implemented as asynchronous dual port memory. The user can connect the read port directly to its own clock domain. The MGT component is configured to work in far-end parallel loopback mode. This mode allows simultaneous data receiving on RX parallel port and data forwarding to the TX parallel port.

2.2 Vivado IP Ports

This chapter describes the ports and their uses.

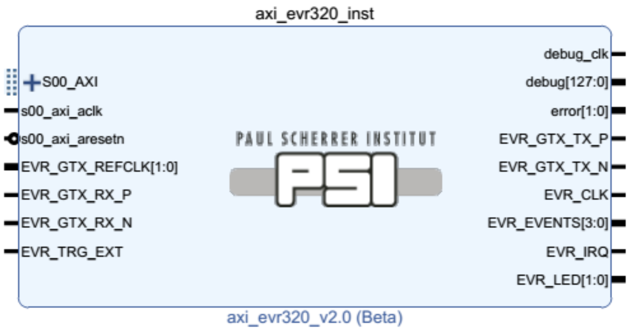

Figure 2: Component Overview

Port name	In/Out	Type	Description
EVR_TRG_EXT	in	std_logic	External trigger input. Could be used for example to connect an external timing input
EVR_GTX_REFCLK	in	std_logic_vector(1:0)	Reference clock 142.8 MHz for MGT. The clock frequency and stability has to be within the range specified by Xilinx for the GTX.
EVR_GTX_RX_P EVR_GTX_RX_N	in	std_logic	Differential serial data RX input for the MGT0. These inputs are used to connect timing input serial stream.
EVR_GTX_TX_P EVR_GTX_TX_N	out	std_logic	Differential serial data TX output for the MGT0. These outputs are used to connect timing output serial stream.
EVR_CLK	out	std_logic	This clock comes from the clock recovery circuit in RX of MGT. It has deterministic phase with respect to the clock source in the EVG.
EVR_EVENTS	out	std_logic_vector(3:0)	User defined events decoder output. The bits generate single clock cycle pulses of o_clk_142MHz8 when the event is received. The pulses are generated when event configured in `evr_params.event_numbers` are detected and only during 10 ms after the last correctly received segmented data buffer frame.
EVR_IRQ	out	std_logic	This signal is set if an event EVR_EVENTS was decoded and triggered some actions.
EVR_LED	out	std_logic_vector(1:0)	These signals are intended for ease of use. They sould indicate to the user of a hardware if the link has loss of sync and if it was triggered in the last 10 ms. EVR_LED (0): denotes the EVR_EVENTS(0) did toggle in the last 10 ms. EVR_LED (1): denotes a loss of sync.
s00_axi_aclk	in	std_logic	User clock to clock ports for evr320 params, status and memory interface. It can have arbitrary frequency.
s00_axi_aresetn	in	std_logic	User asynchronous low active reset.
s00_axi	in/out	various	This is the AXI4 bus interface.
debug_clk	out	std_logic	Developer ILA interface clock.
debug	out	std_logic_vector(127:0)	Developer ILA interface data.

Table 1: Port description of evr320_v2.0

2.3 Configuration

The component starts working automatically whenever the timing signal is connected the RX input. Only the user events table has to be configured to detect the required events in run time.

2.3.1 Control and Status Register Map:

Address	Size	Access	Description
0x00000000	32bit	RO	MGT status vector: bit[0] – GTX PLL lock detected bit[1] – RESETDONE bit[8] – LOSSOFSYNC bit[9] – RESETDONE (for legacy reasons with old EVR)
0x00000004	32bit	RW	User events codes: bit[0:7] – user event 0 code bit[8:15] – user event 1 code bit[16:23] – user event 2 code bit[24:31] – user event 3 code
0x00000008	32bit	RW	MGT control vector: bit[0] – Reset GTX. This will completely reset the GTX component.
0x0000000C	32bit	RW	User events codes: (not implemented for ifc1210) bit[24] – ‘0’ := Use decoded MGT events / ‘1’ := Use external LVDS trigger
0x00000010	32bit	RW	Decoder enable on receiving event: bit[0] – ‘1’ := enable decoding of user event 0 code bit[8] – ‘1’ := enable decoding of user event 1 code bit[16] – ‘1’ := enable decoding of user event 2 code bit[24] – ‘1’ := enable decoding of user event 3 code
0x00000014	32bit	RW	reserved for -Force Event-
0x00000018	32bit	R	reserved for –Implementation Options-
0x0000001C	32bit	R	Recovered Clock Frequency [Hz] (ifc1210 only)
0x00000020	32bit	RW	Minimum number of correctly received segmented data buffer frames necessary in order to allow triggering on events. (default: 100/0x64)
0x00000024	32bit	RW	Minimum time with correctly received segmented data buffer frames necessary in order to allow triggering on events. (default: 0x15CA20)
0x00000030	32bit	RW	Latency Measurement - Config bit[7:0] – event number
0x00000034	32bit	R	Latency Measurement Counter value since ‘event number’ detected
0x00000038	32bit	R	Latency Measurement Bit[0] – Arm Counter: Write ‘1’ to arm counter again
0x0000003C	32bit	R	Reserved for Latency Measurement
0x00000040	32bit	RW	Event Recorder - Control: bit[0] – ‘1’ := enable decoding of sos event (start-of-sequence) bit[8:15] – sos event code (default: 0x20/32)
0x00000044	32bit	R	Event Recorder - Read Handshake: bit[0] – ‘1’ := data valid in buffers bit[8] – ‘1’ := data error on readout (recommendation: discard data) bit[16] – ‘1’ := data read acknowledge (usage: write ‘1’ after buffer read) bit[24] – ‘1’ := error acknowledge (usage: write 1’ to clear data error flag)
0x00000048	32bit	R	Event Recorder - User Event Counter Events: 0x01-0x6F and 0x80-0xFF (usage: counter value defines how many entries in event recorder buffers are valid)
0x00000050	64bit	RW	User event pulse delay parameters in recovery clock cycles bit[15:0] – user event 1 pulse delay parameter bit[31:16] – user event 2 pulse delay parameter bit[47-32] – user event 3 pulse delay parameter

			bit[63:48] – user event 4 pulse delay parameter
0x00000058	64bit	RW	User event pulse width parameters in recovery clock cycles bit[15:0] – user event 1 pulse width parameter bit[31:16] – user event 2 pulse width parameter bit[47-32] – user event 3 pulse width parameter bit[63:48] – user event 4 pulse width parameter
0x00000060	32bit	RW	SOS pulse delay & width parameter bit[15:0] – sos pulse width parameter bit[31:0] – sos pulse delay parameter

Table 2: Control and Status Register Address map of evr320

2.3.2 Segmented Data Buffer Map:

Address	Size	Access	Description
0x00000080- 0x0000087F	2KB	R	Segmented data buffer
0x00000880- 0x0000107F	2KB	R	Segmented data buffer synced with user event 0
0x00001080- 0x0000187F	2KB	R	Segmented data buffer synced with user event 1
0x00001880- 0x0000207F	2KB	R	Segmented data buffer synced with user event 2
0x00002080- 0x0000287F	2KB	R	Segmented data buffer synced with user event 3

Table 3: Segemented Data Buffer Address map of evr320

2.3.3 Event Recorder Map:

Address	Size	Access	Description
0x00002880- 0x0000307F	2KB	R	Segmented data buffer synced with SOS
0x00003080- 0x0000347F	1KB	R	Event Numbers Timestamp (sorted in time domain, 32bit aligned) bit[0:31] – event code timestamp [clock cycles]
0x00003480- 0x0000367F	512B	R	Segment Timestamps from data buffer (32bit aligned) bit[0:31] – Segment Start timestamp [clock cycles]
0x00003680- 0x0000377F	256B	R	Event Numbers (sorted in time domain, Byte aligned) bit[0:7] – event code 1st (typical SOS code) bit[0:7] – event code 2nd bit[0:7] – event code 3rd : bit[0:7] – event code 255th
0x00003780- 0x0000387F	256B	R	Event Flags (sorted by event code, Byte aligned) bit[0] – event code 0 bit[0] – event code 1 bit[0] – event code 2 : bit[0] – event code 255

Table 4: Event Recorder Address map of evr320

2.4 Event Recorder

Idem to the basic EEVR events, the Event Recorder triggers onto a configurable event (Start-of-Sequence Event or SOS-Event, default 32/0x20) when the Enable Bit is set (default off='0').

2.4.1 Data Readout

To guarantee valid data on readout the following rules must be followed.

- Minimum number of correctly received data buffer frames must set >= 1
- Minimum time of correctly received data buffer frames must be >= 0x15CA20 for SwissFEL@100Hz with 142.8MHz reference clock
- The software needs to readout the desired data and confirm the termination with sending the *Read Ack* before the next *SOS Event*

Two typical sequences are shown in Figure 3 and Figure 4 with the separate Acknowledge signals for each status flag.

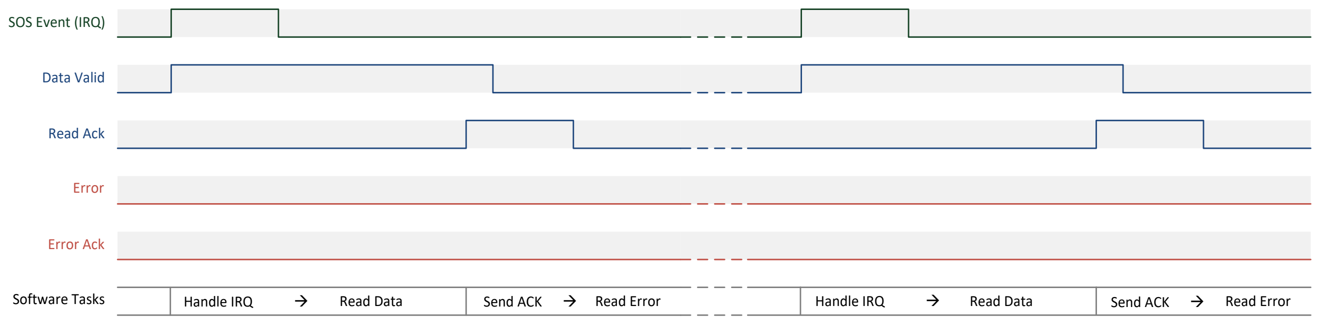

Figure 3: Normal Readout Sequence

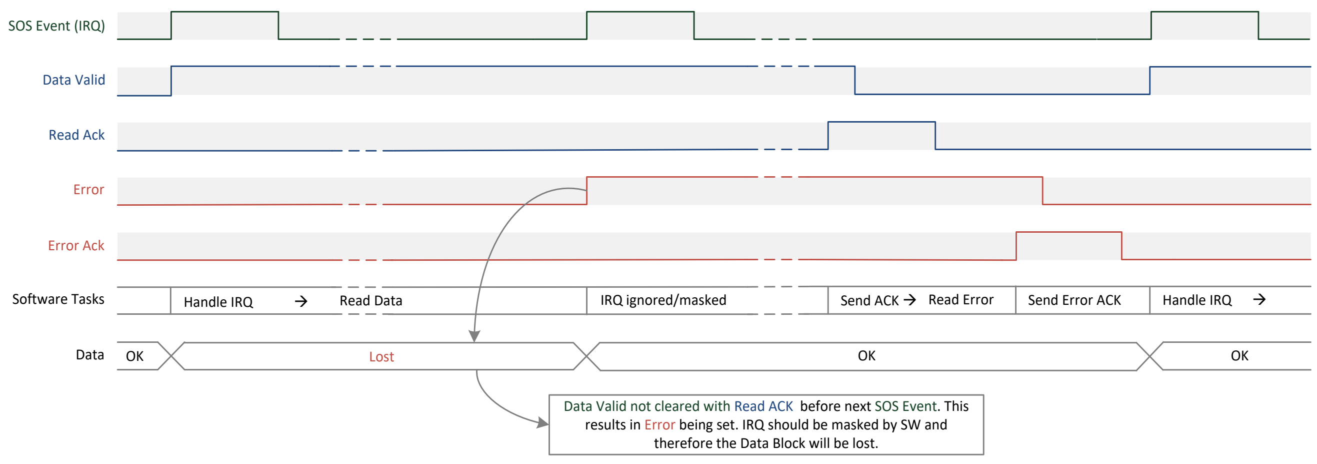

Figure 4: Erroneous Readout Sequence

2.4.2 Data Validation

The *User Events Counter* register represents the amount of valid entries in the memory blocks *Event Numbers* and *Event Numbers Timestamp*.

2.5 Design constraints

The design needs to specify the timing of the MGT output and all logic used with by the 142.8 MHz recovered clock.

2.6 Latency Measurement

The latency measurement provides a counter to support software performance diagnosis.

- Counting clock cycles (8ns) up to 10ms and stop
- Start counting when counter is armed and a specific event (Reg 0x30) detected
- The software reads the current counter value either from register 0x34
- The counter is cleared and rearmed only when the software writes ‘1’ to register 0x38

2.7 IFC1210 Wrapper

The embedded EVR component can also be used on ifc1210 board, therefore a wrapper has been made and here below once can find its port map description and architecture. It consists in having added a TMEM interface (cf. Tosca2 documentation) with new features such as the extension of the pulse length and a settable delay for each output event (cf. §2.3.1). The configuration vector length has been set to 16 bits.

2.7.1 Generics

g_MGT_LOCATION “GTXE1_X0Y0” -> “GTXE1_X0Y11” | “GTXE1_X016” -> “GTXE1_X0Y19”
g_FACILITY “SFEL” | “HIPA” select the facility where IOC is installed
g_EVENT_RECORDER true = enable Event recorder functionality
false = disable Event recorder functionality
g_XUSER_CLK_FREQ Clock frequency in Hz
g_EVT_HOLDOFF table value for minimum of clock cycles between input pulses event (sos.0,1,2,3,4)

2.7.2 Interfaces

Signal	Direction	Width	Description
Debug interface			
debug_clk	Output	1	Chipscope Clock
debug	Output	128	Chipscope ICON/ILA purpose
Tosca 2 TMEM			
xuser_CLK	Input	1	Clock xuser (125MHz typical)
xuser_RESET	Input	1	Reset xuser (active high)
xuser_TMEM_ENA	Input	1	Bus access enable TMEM (active high)
xuser_TMEM_WE	Input	8	Bus Write enable TMEM
xuser_TMEM_ADD	Input	11	Bus Address TMEM
xuser_TMEM_DATW	Input	64	Bus data write TMEM
xuser_TMEM_DATR	Output	64	Bus data read TMEM
MGT			
mgt_refclk_i	Input	1	Reference clock for MGT Rx
mgt_sfp_los_i	Input	1	SFP Loss of signal (light on receiver)
mgt_rx_n	Input	1	MGT RX negative pin
mgt_rx_p	Input	1	MGT RX positive pin
mgt_tx_n	Output	1	MGT TX negative pin
mgt_tx_n	Output	1	MGT TX positive pin
mgt_status_o	Output	32	MGT status output

mgt_control_i	Input	32	MGT control input
User interface MGT clock			
clk_evr_o	Output	1	Recovery clock
usr_events_o	Output	4	Events – trigger with one clock cycle length
sos_events_o	Output	1	Start of sequence of the events
usr_events_adj_o	Output	4	Events – trigger adjusted with pulse length and delay
sos_events_adj_o	Output	1	Start of sequence of the events adjusted with pulse length and delay
Decoder AXI stream			
stream_clk_i	Input	1	AXI stream clock – user clock
stream_dat_o	Output	8	AXI stream data output
stream_addr_o	Output	11	AXI stream address output
stream_valid_o	Output	1	AXI stream valid output

2.7.3 Architecture

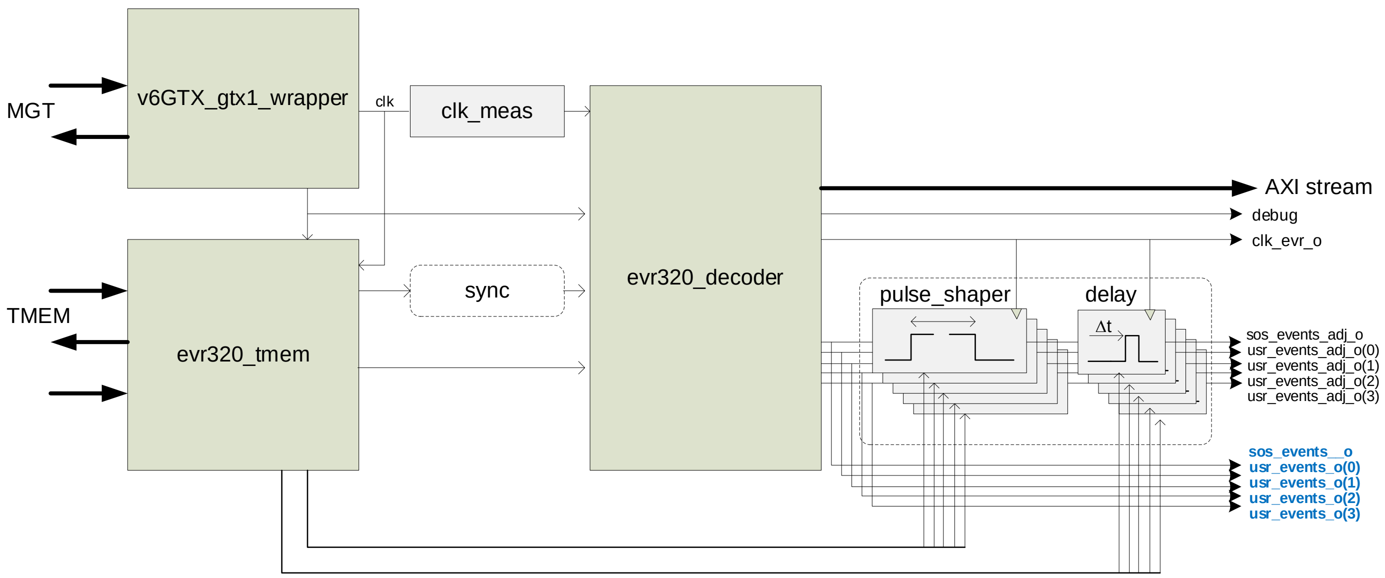

Figure 5: IFC1210 wrapper architecture