Complete MRF Timing System on MicroTCA.4

MTCA/ATCA Workshop for Research and Industry

IHEP, Beijing, June 2019

Jukka Pietarinen

Micro-Research Finland Oy

MRF Timing System Development

- The MRF Timing system originally developed for the Swiss Light Source (SLS), Paul-Scherrer Institut in 1999
- First commercially available timing system for particle accelerators
- Current users include
  - SLS, PSI
  - Diamond Light Source Ltd., U.K.
  - SSRF, Shanghai, China
  - Australian Synchrotron
  - ALBA, Spain (Tango)
  - Elettra, Trieste, Italy (Tango)
  - BEPCII, Insitute for High Energy Physics, Beijing, China
  - LCLS, Stanford Linear Accelerator Center, USA
  - KEK, Japan
  - FERMI, Trieste, Italy (Tango)
  - TLS, Taiwan
  - NSLS II, Brookhaven, USA
  - PAL-XFEL, Pohang, South Korea
  - LANSCE upgrade, Los Alamos, USA
  - FRIB, Michigan, USA
  - MAX IV, Sweden
  - SwissFEL, PSI
  - STFC, Daresbury
  - ESS, Sweden
- Most of the sites listed above are using EPICS

What do you need a timing system for?

- Phase and frequency locked clocks at different locations
- Trigger something at different locations at exactly the same time
- Make something happen in sequence with predefined time intervals
- Synchronize the local time at different locations with high precision
- Timestamp events at different locations to analyze what happened first

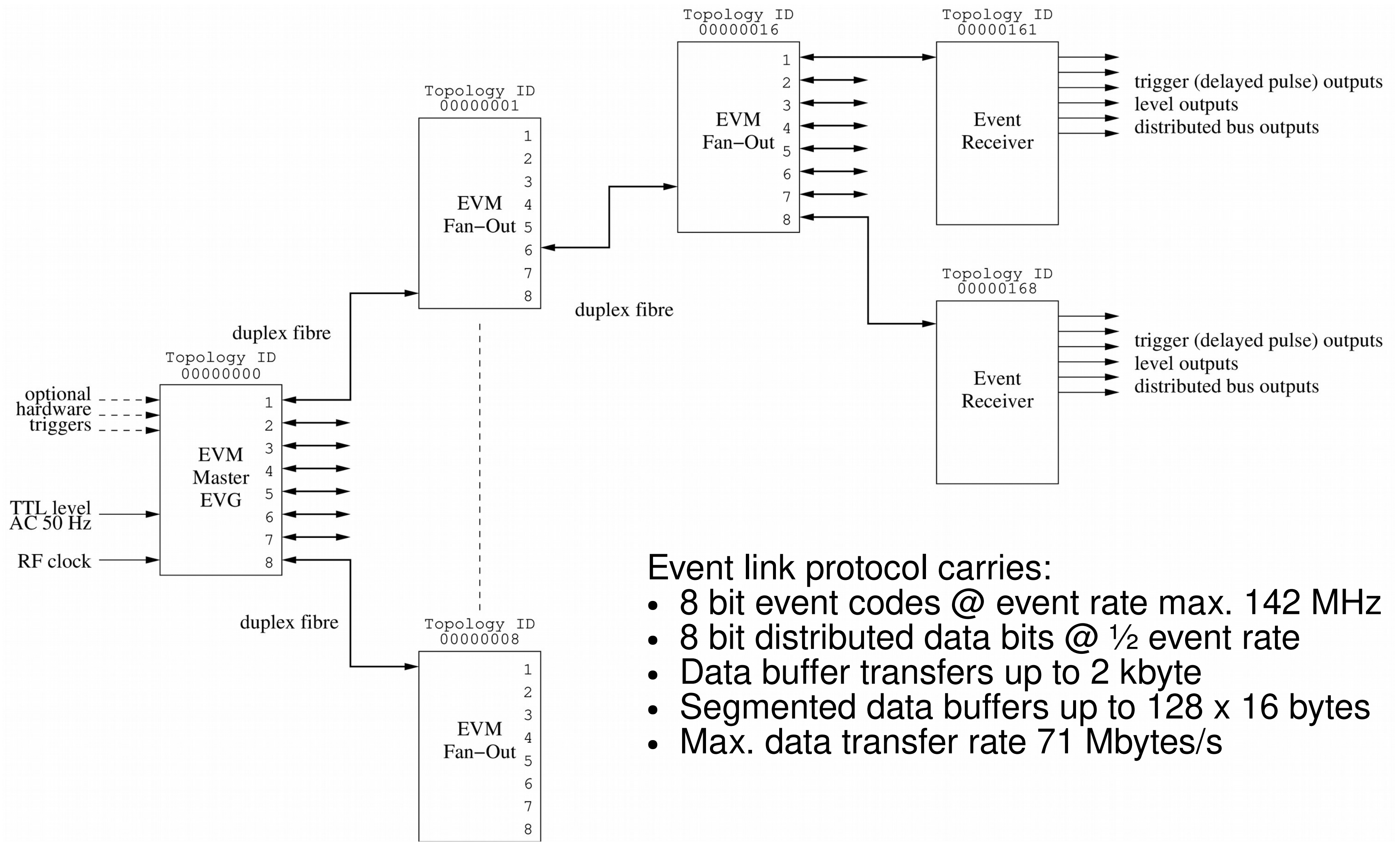
Operating Principle of the MRF Timing System

- Event based system
  - 256 Event codes, one event code/cycle (7 ns - 20 ns or 50 MHz to 142 MHz)
- In addition to events there are eight signals (distributed bus) that are updated at half the rate (14 ns - 40 ns or 25 MHz to 71 MHz)
- Synchronous data transfers
- Transmission medium: optical fiber
  - typically multi-mode, OM3 or better up to 550 m
  - single-mode, up to 10 km
- Line code or “event stream” based on 8B10B encoding

Event Clock

- Base clock of MRF timing system
- Typically driven from external source e.g. accelerator RF signal
- Has to be in range 50 MHz to 142 MHz (7 ns to 20 ns cycle)
- Has to be provided to the Timing System Master, all other nodes regenerate this clock from the event stream received through the fiber
- The Timing Master has an input divider to divide the provided RF clock by 1, 2, 3, ..., 12, 14, 15, 16, ..., 32.

Typical System Layout - Delay Compensation (DC)



Event link protocol carries:

- 8 bit event codes @ event rate max. 142 MHz
- 8 bit distributed data bits @ ½ event rate
- Data buffer transfers up to 2 kbyte
- Segmented data buffers up to 128 x 16 bytes
- Max. data transfer rate 71 Mbytes/s

EVG Basic Building Blocks

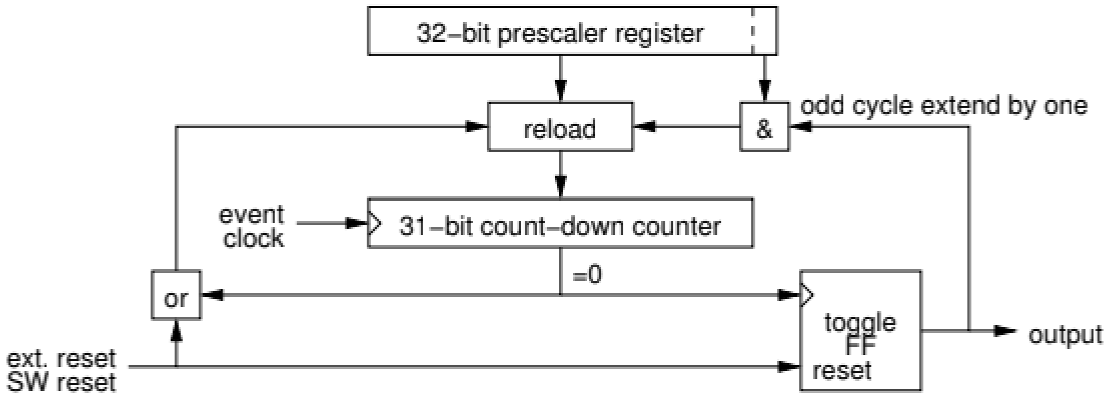
- Trigger Events
- Multiplexed Counters
- Sequencers
- Event Analyzer
- Software Event
- Time distribution logic
- Configurable Size Data buffer
- Segmented Data buffer

Trigger Event

- A Trigger Event has one event code assigned to it
- Upon a trigger this event code is sent out
- Sources for triggers:
  - External signal
  - Multiplexed Counter edge
  - AC main synchronization logic

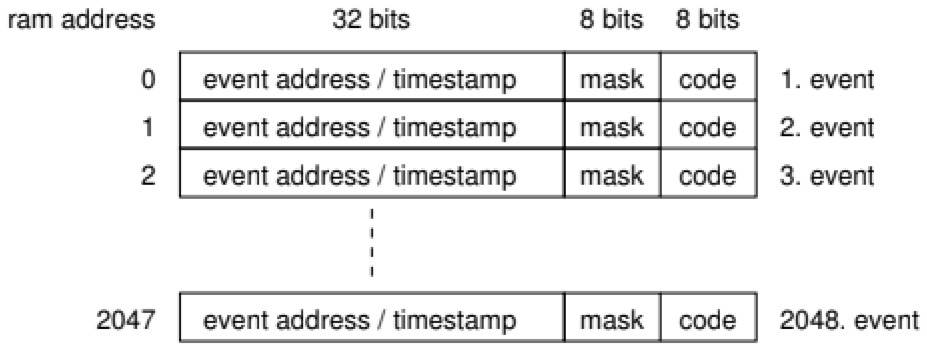
Multiplexed Counter

- 32 bit counters running at the event clock rate
- Can produce frequencies from event clock rate/(2^{32} - 1) to event clock rate/2
- Counters can be reset simultaneously by software
- Configurable polarity



Sequencer

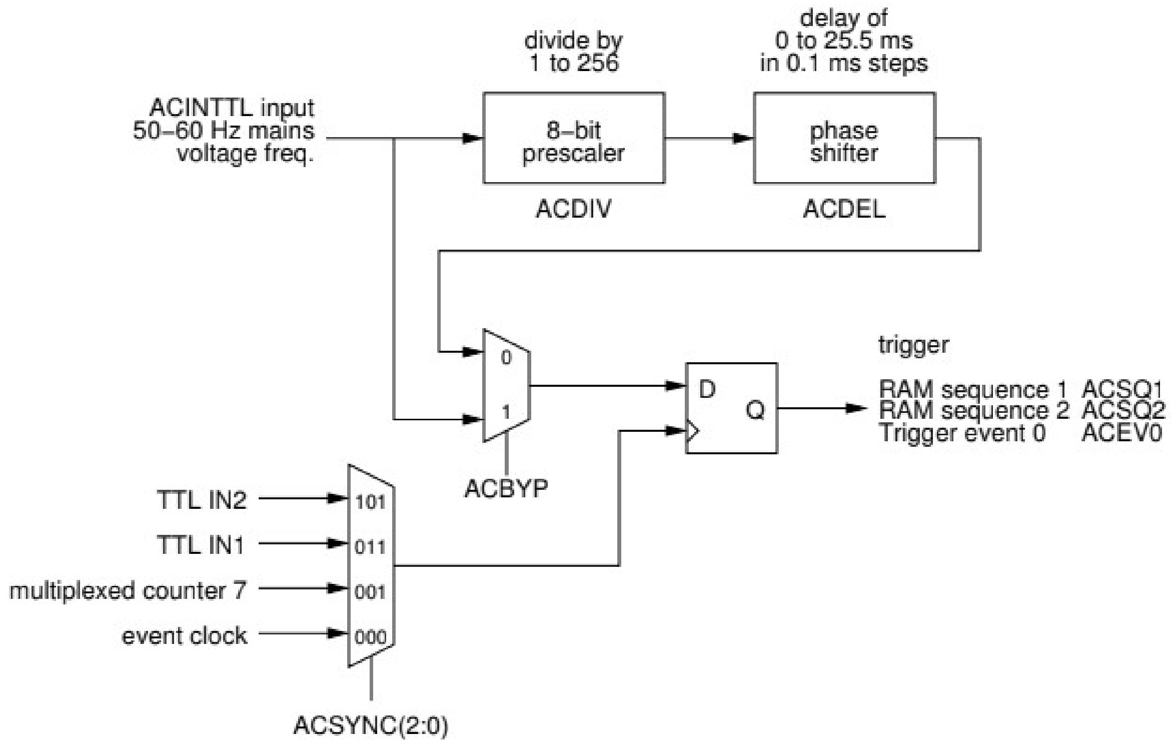
- A sequencer sends out events in predefined order with predefined timing
- 2k dual-port memory for 32-bit timestamp, event code and mask
- Operating modes: single, normal (trigger) and recycle



ram address	32 bits	8 bits	8 bits	
0	event address / timestamp	mask	code	1. event
1	event address / timestamp	mask	code	2. event
2	event address / timestamp	mask	code	3. event
2047	event address / timestamp	mask	code	2048. event

Sequencer triggering

- Multiplexed counter
- AC Synchronization logic
- External trigger
- Software trigger



Sequencer operating modes

- Single mode
  - Plays back sequence once until end sequence event
- Normal mode
  - Plays back sequence and waits for next trigger after end sequence event
- Recycle mode
  - Plays back sequence and immediately restarts sequence from beginning after end sequence event

Event Analyzer

- Tools to record all events sent out by the EVG
- 64-bit counter counting event clock cycles
- 2k memory to store event codes, 64-timestamp and distributed bus state

Sequencer and Event Analyser Example

address	timestamp	event	
0	0	0x20	1. Event
1	85168*66	0x2a	2. Event
2	94537*66	0x24	3. Event
3	94538*66	0x25	4. Event
4	94821*66	0x2c	5. Event
5	282002*66	0x30	6. Event
6	283895*66	0x3c	7. Event
7	284000*66	0x7f	End sequence

- 264/4 = 66, 528 us cycles
- Line sync. divider 10
- 50 Hz applied to line sync. Input
- Trigger event enabled to send 0x11 on seq. trigger

Event Analyser with 64-bit time counter

Analyser time (s)	Offset (us)	code
0,190072274	-1,07	11
0,190073347	0,00	20
0,235073191	44999,84	2a
0,240023448	49950,10	24
0,240023977	49950,63	25
0,240173504	50100,16	2c
0,339073511	149000,16	30
0,340073707	150000,36	3c
0,390076629	-1,07	11
0,390077702	0,00	20
0,435077546	44999,84	2a
0,440027803	49950,10	24
0,440028332	49950,63	25
0,440177859	50100,16	2c
0,539077866	149000,16	30
0,540078062	150000,36	3c

Time Distribution Logic

- Support for 32 bit Seconds register
- 32-bit sub second register - e.g. 1 us precision can be used
- Seconds value is send out with event codes
- Sub-second clock is send out as an event or on the distributed bus
- 1 PPS and faster clock e.g. 1 MHz have to be supplied externally and these two clock must be rising edge aligned

Configurable Size Data Buffer

- 2k byte dual-port memory
- Transfer size 4 bytes to 2k in 4 byte increments
- Controlled by software

Segmented Data Buffer

- Introduced with Delay Compensation hardware
- 2k byte dual-port memory
- 128 16-byte segments
- Can send a single or multiple segments at a time
- Controlled by software
- Last segment #127 is reserved for delay compensation
  - delay compensation data
  - Topology ID

EVR Basic Building Blocks

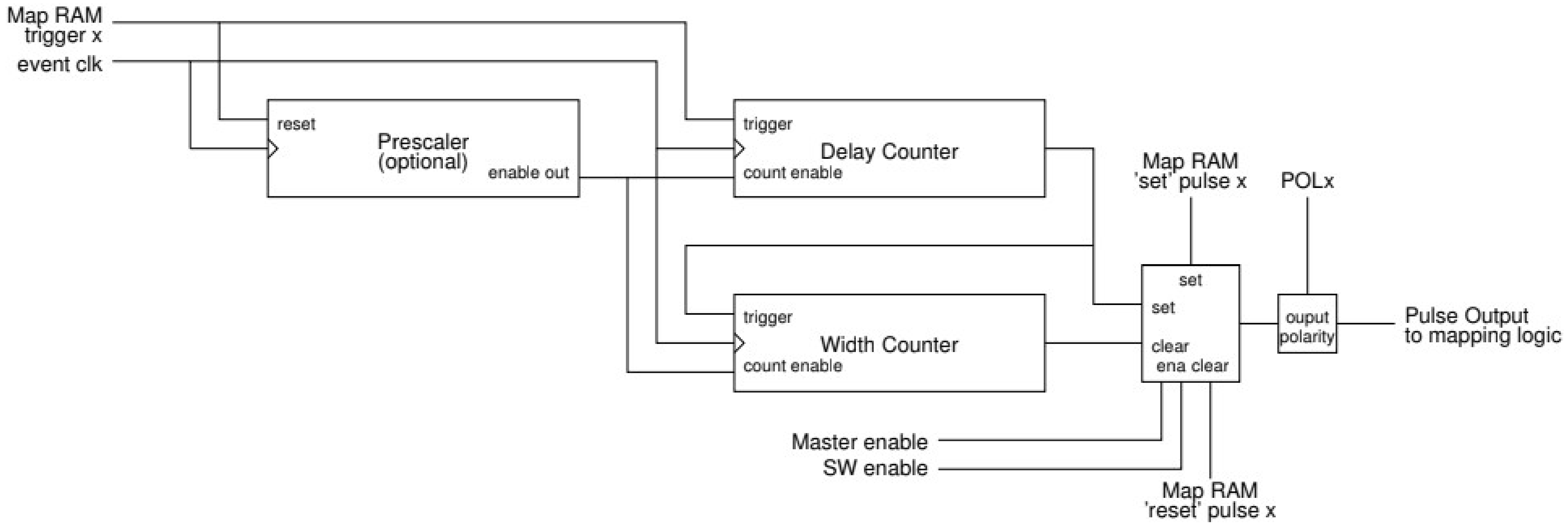
- Event Mapping RAM
- Pulse Generators
- Prescalers
- Timestamping logic
- Event FIFO
- Event Log
- Data buffer
- Segmented Data buffer

Event Mapping RAM

- 256 x 128 bit dual-port RAM
- Each event code has 128 bits assigned and these bits define what functions are to be performed

Map bit	Default event code	Function
127		Save event in FIFO
126		Latch timestamp
125		Led Event
124		Forward event
123	0x79	Stop log
122		Log event
101	0x7A	Heartbeat
100	0x7B	Reset Prescalers
99	0x7D	Timestamp reset event
98	0x7C	Timestamp clock event
97	0x71	Timestamp Seconds ‘1’
96	0x70	Timestamp Seconds ‘0’
95 - 64		Trigger pulse generator 31 - 0
63 - 32		Set pulse generator 31 - 0 output high
31 - 0		Reset pulse generator 31 - 0 output low

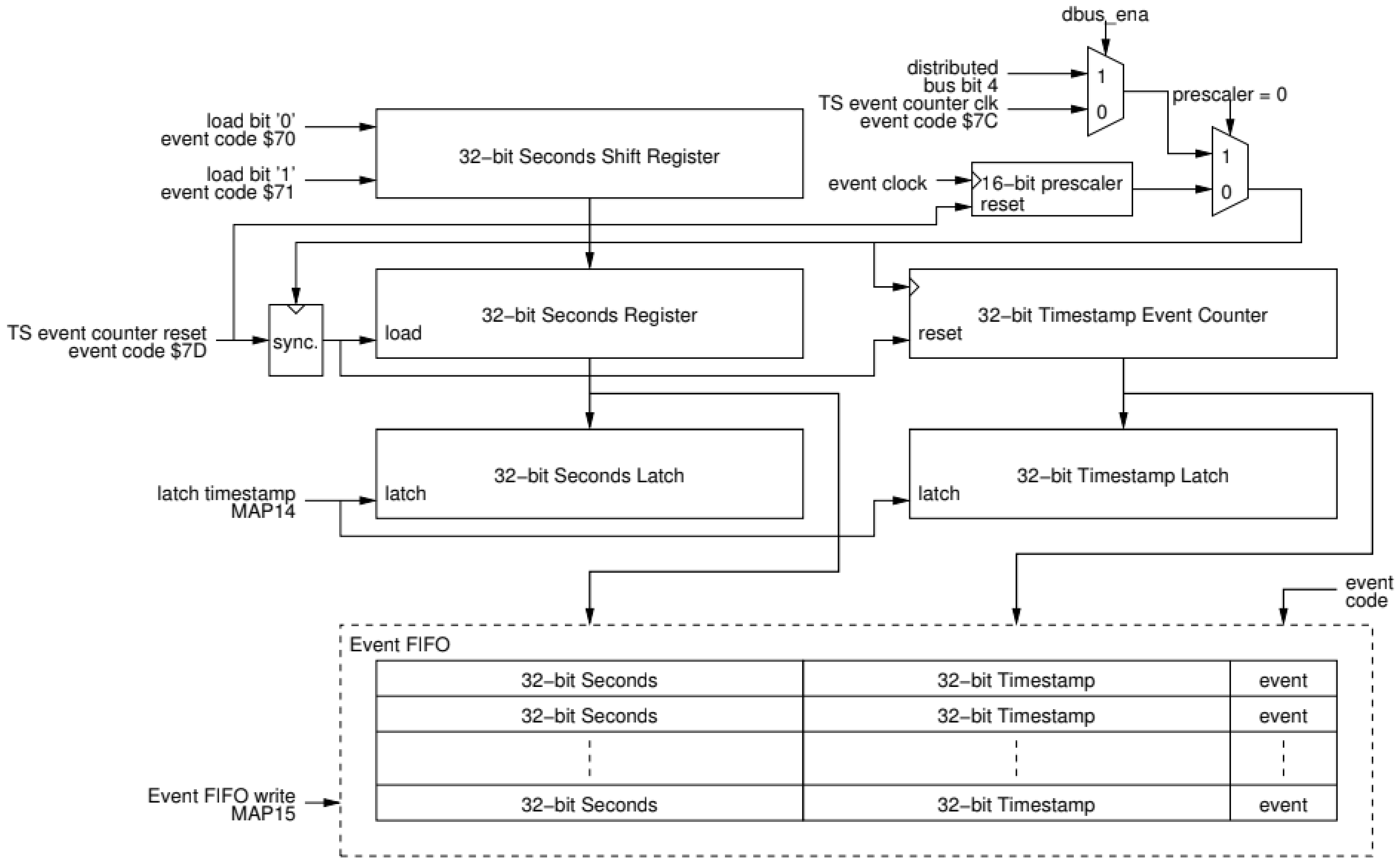
Pulse Generator



Prescalers

- Programmable counter that can produce different frequencies
- All prescalers are reset by one event code, thus all prescalers on all EVRs can be synchronized
- Can be output directly on I/O or used to trigger pulse generators
- Recent change: addition of phase control
  - When the ‘reset prescalers’ event is received the prescaler counter value is reloaded from the phase offset register

Timestamping Logic and Event FIFO



Event Log

- Similar to Event FIFO, but with ring buffer
- Event FIFO stops when full
- Event Log rolls over and overwrites old events
- Can be used for diagnostics: stop log event e.g. after beam loss, read out log events

Data Buffer

- Receive Data Buffer send by EVG
- Data buffer has to be armed to receive anything
- Data buffer is automatically disable after a received buffer
- Data buffers can be lost if buffer is not read and re-armed before EVG sends next buffer
- Software interrupt upon reception

Segmented Data Buffer

- Receive Segments send by EVG
- Segmented Data buffer is enabled all the time
- Flags for each segment: receive complete, checksum mismatch, overflow flag
- Each segment has a received data bytes counter
- Software interrupt configurable for each segment independently

Flip-Flop Outputs (recent addition)

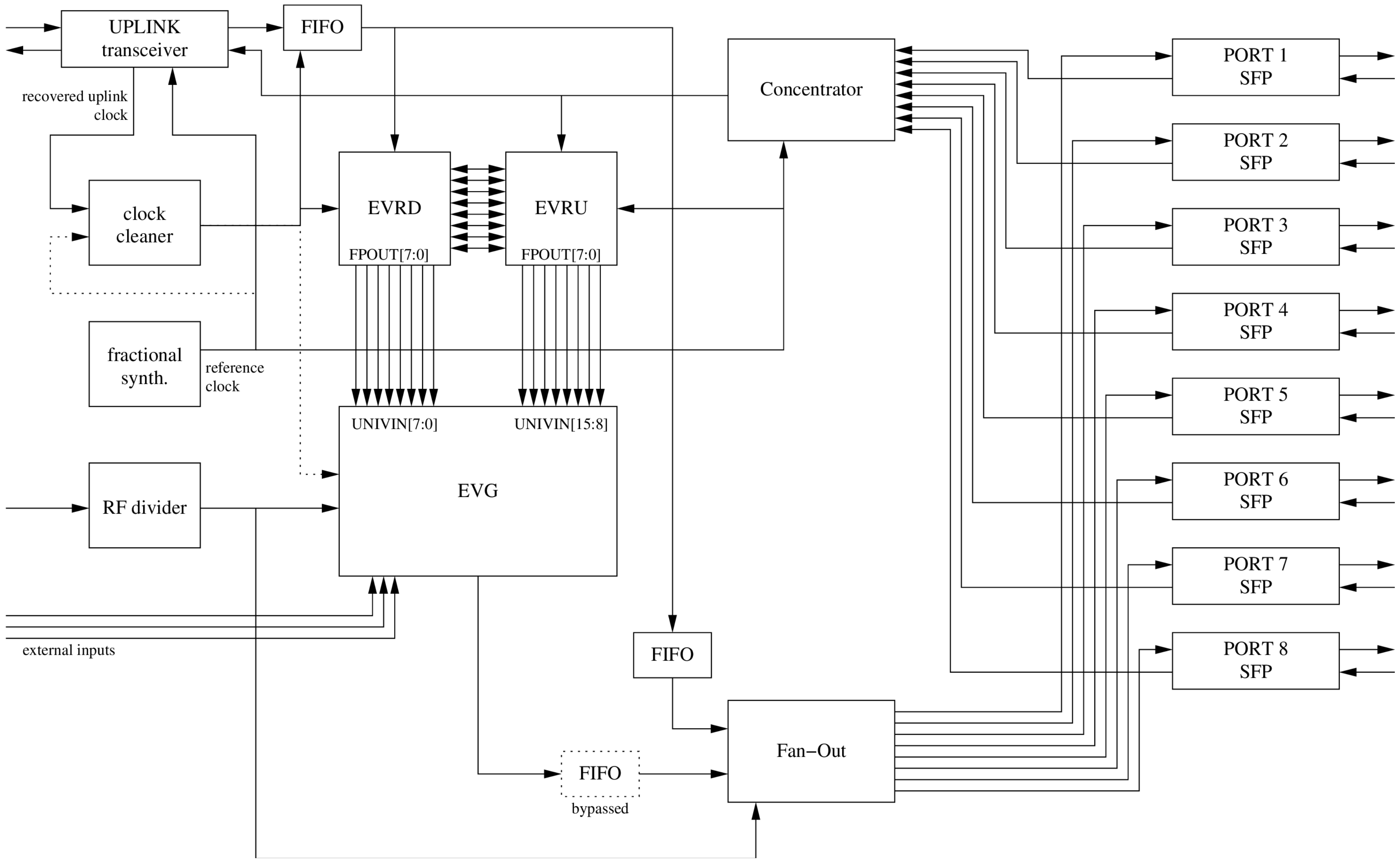
- Output stage that is using two pulse generators to control a set/reset flip-flop
- Flip-flop 0
  - high level on pulse generator 0 sets output high
  - high level on pulse generator 1 resets output low
- Flip-flop 1
  - high level on pulse generator 2 sets output high
  - high level on pulse generator 3 resets output low
- Etc.
- If both PGs are high → output low

EVM, EVG, EVR

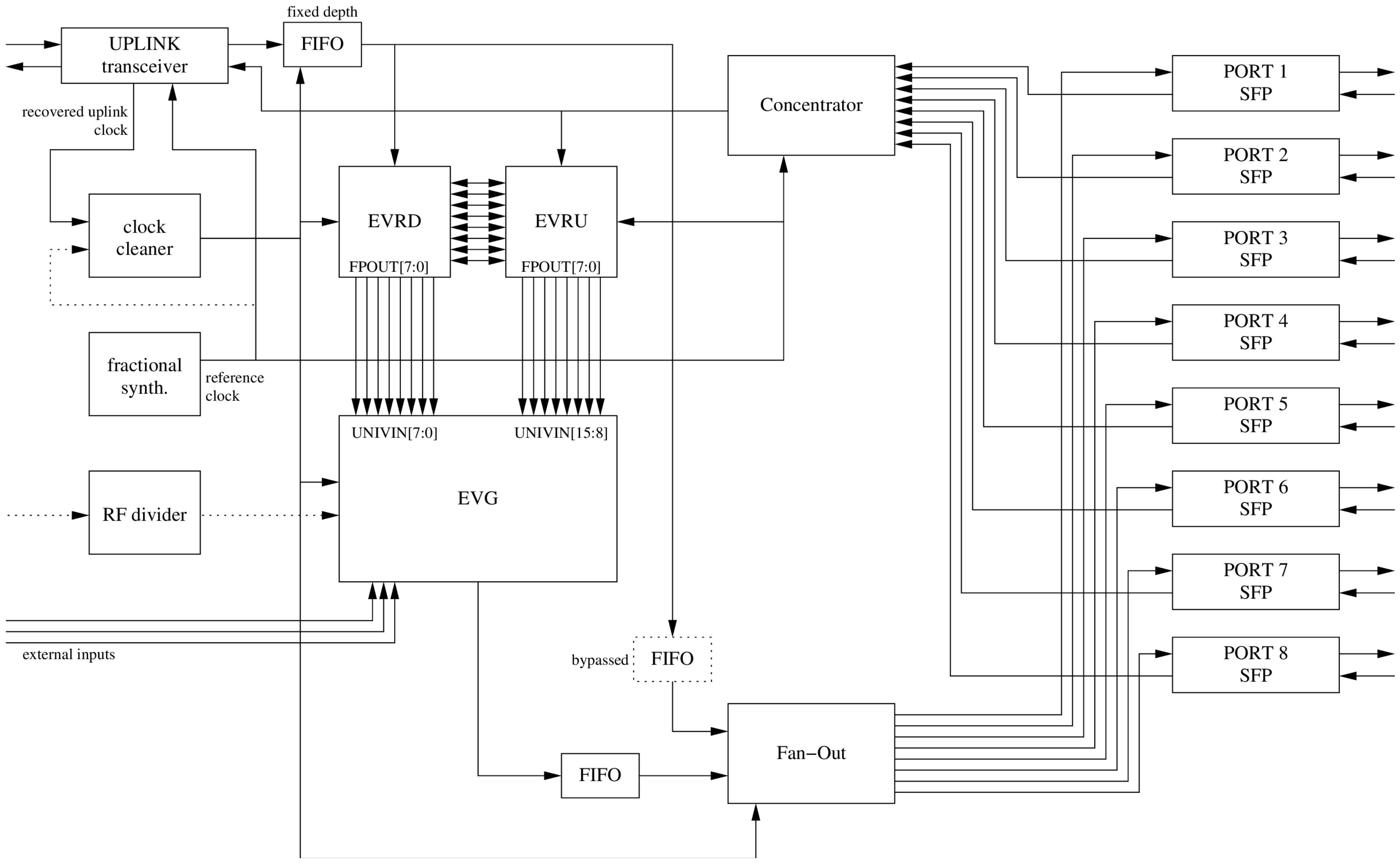
To make it even more confusing...

- The difference between Event Generator and Event Receiver is not that clear anymore as it used to be
- EVG features have been added to EVRs and vice versa

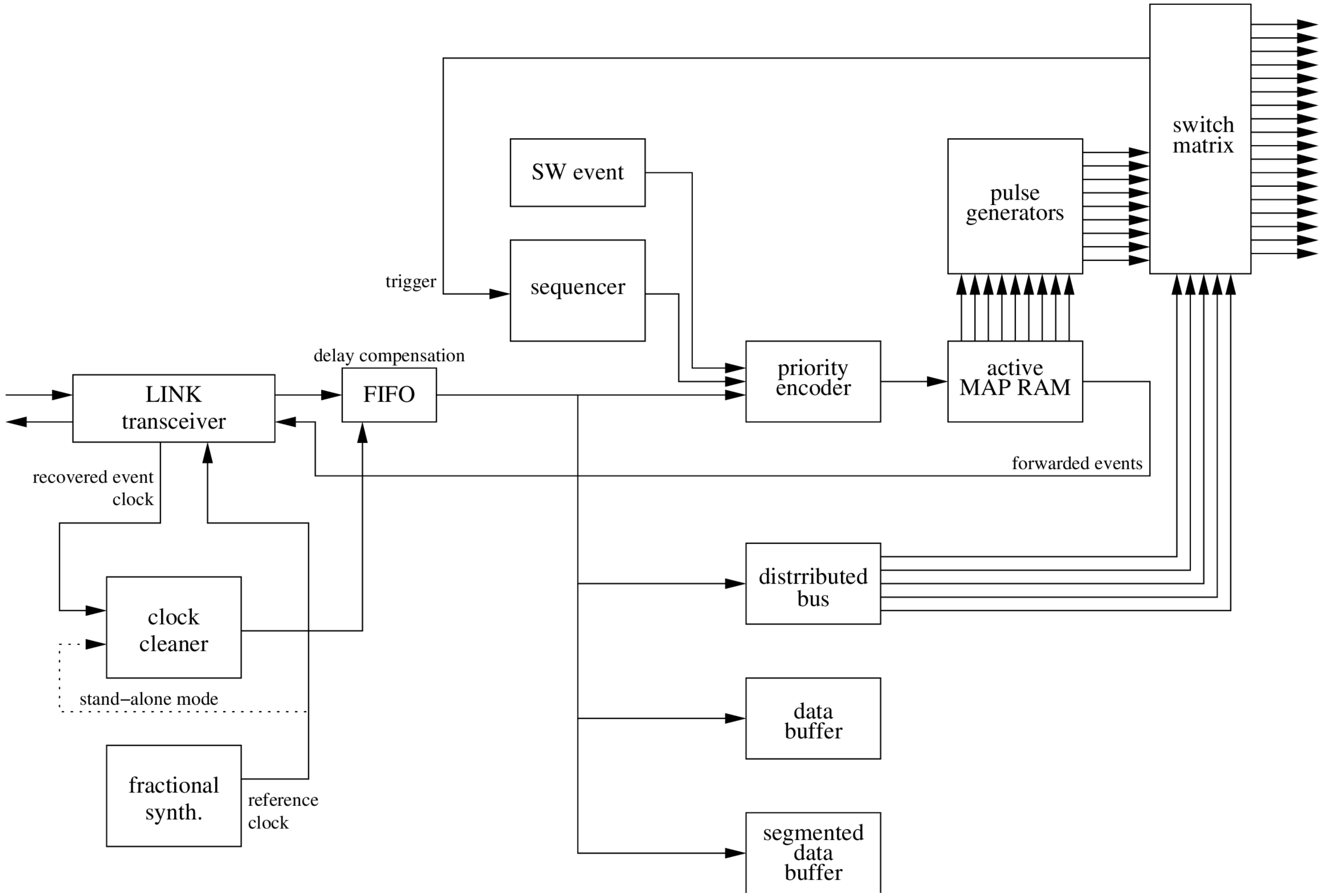
EVM configured as System Master



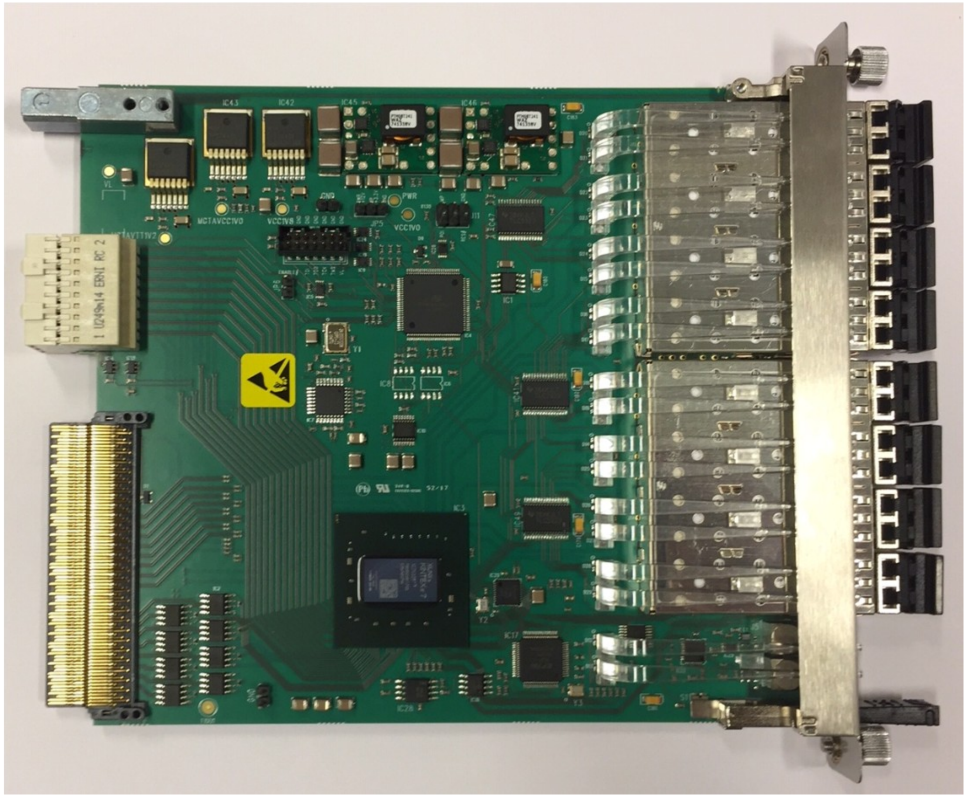
EVM configured as Fan-Out



EVR Simplified Block Diagram



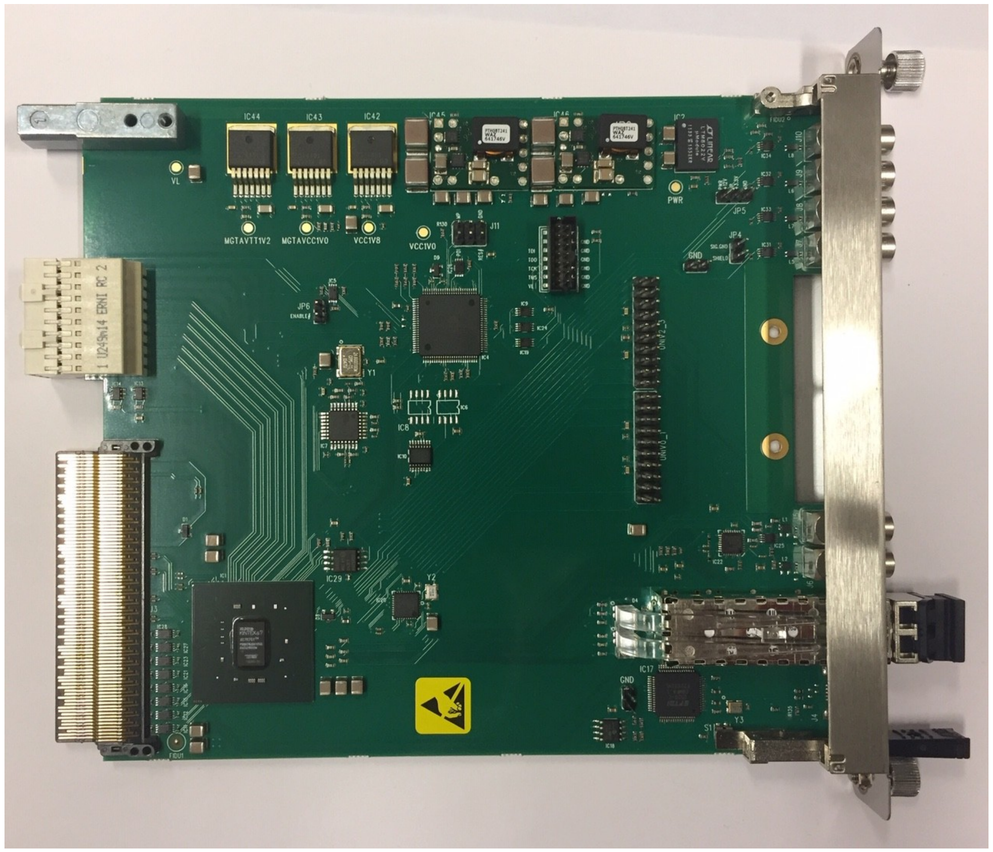
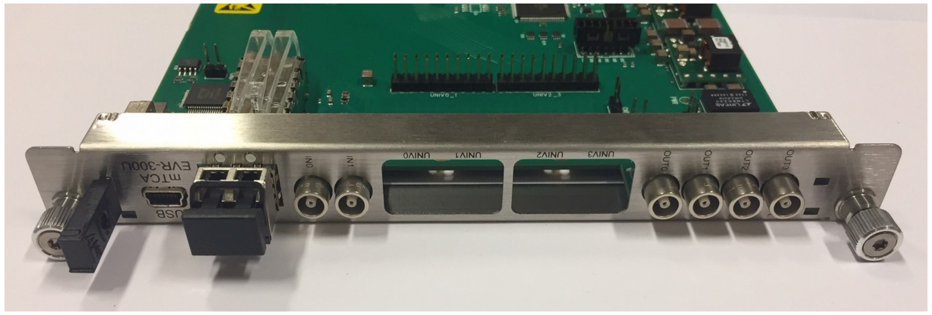
mTCA-EVM-300



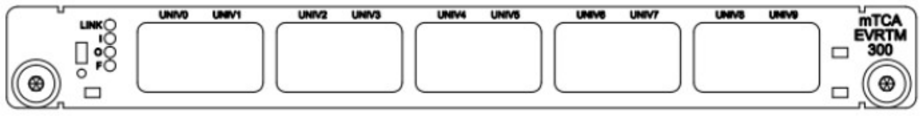
- Event Generator (EVG)
  - 2 sequencers with maskable events
    - Max. 2047 events/sequence
    - 32 bit timestamp
  - 8 multiplexed counters
  - data buffer up to 2k bytes
  - segmented data buffer
    - 127 segments, 16 bytes each
- 7-Way Fan-Out/Concentrator
  - +4 backplane ports
- Two Event Receivers (EVR)
  - 8 internal pulse outputs
  - one sequencer
- Event rate conversion (with some data buffer related limitations)
- Front panel input phase monitoring/select features
- Distributed bus phase selection
- RF input monitoring

mTCA-EVR-300U

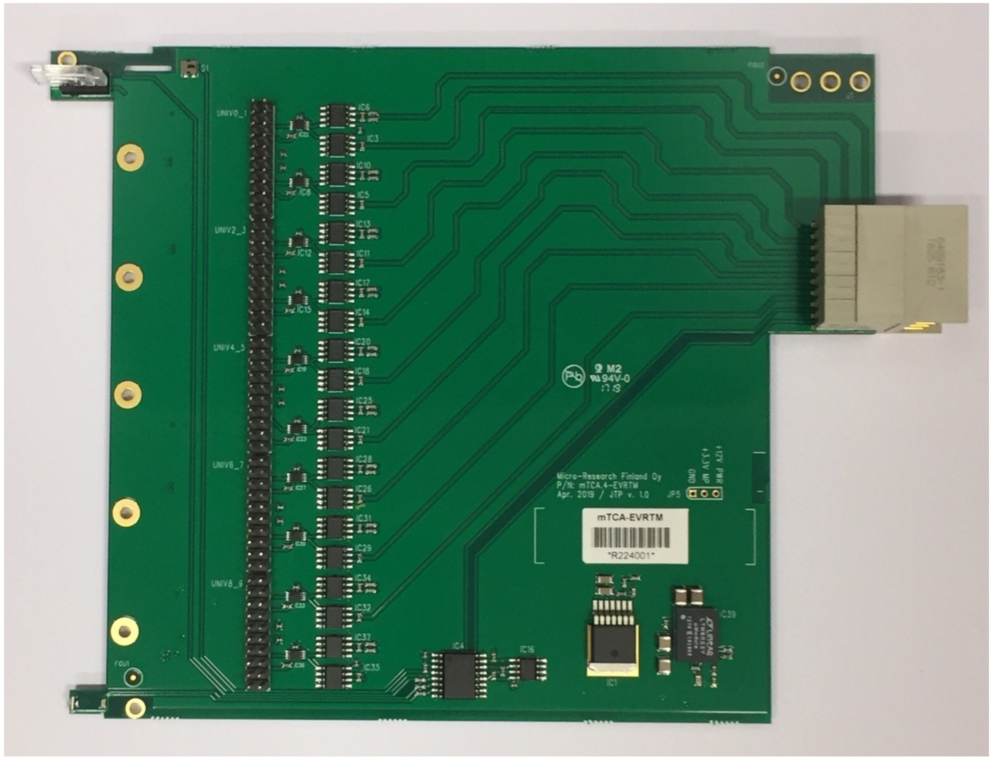
- Event Receiver (EVR)
- Two Universal I/O slots
  - both compatible with -DLY modules
- Four LVTTL outputs
- Two TTL inputs for external triggers
- Backplane triggers
- Can drive TCLKA/TCLKB clocks with GTX logic
- RTM interface, no RTM designed yet

mTCA-EVRTM-300 in development



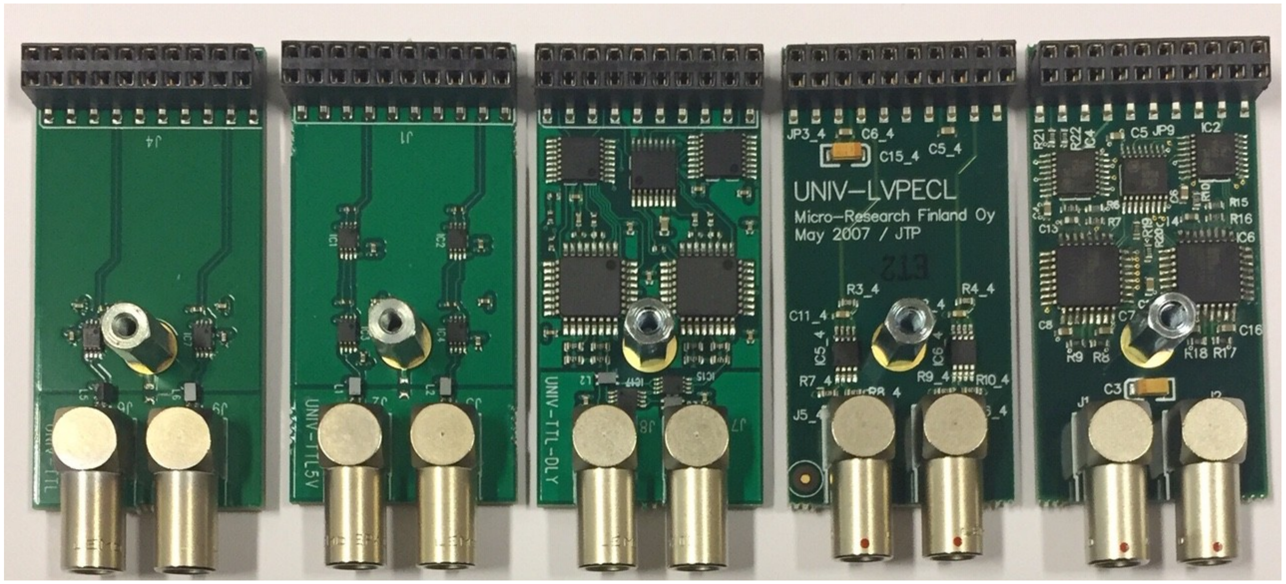
- Rear Transition Module (RTM) for mTCA-EVR and mTCA-EVM
- Five Universal I/O Modules
- Support for -DLY modules in all slots



Universal I/O Modules

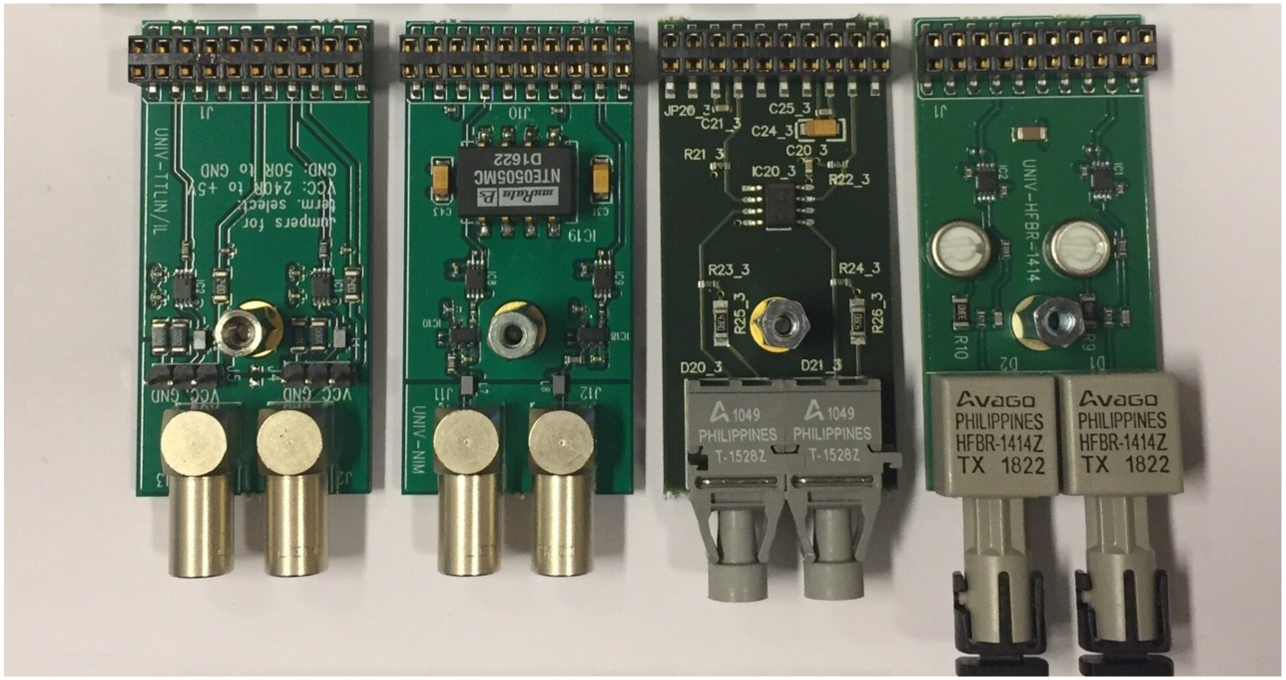
- UNIV-TTL
  - LVTTL output
- UNIV-TTL5V
  - 5V TTL output
- UNIV-TTL-DLY
  - LVTTL output
  - Delay tuning 1024 steps of ~9 ns
- UNIV-LVPECL
  - differential LVPECL output
- UNIV-LVPECL-DLY
  - differential LVPECL output
  - Delay tuning 1024 steps of ~9 ns

UNIV-TTL UNIV-TTL5V UNIV-TTL-DLY UNIV-LVPECL UNIV-LVPECL-DLY



Universal I/O Modules

UNIV-TTLIN/IL UNIV-NIM UNIV-HFBR-1528 UNIV-HFBR-1414



- UNIV-TTLIN
  - TTL input
  - configurable term.
    - 240 ohm to +5V
    - 50 ohm to GND
- UNIV-NIM
  - NIM output
- UNIV-HFBR-1528
  - Optical Output
- UNIV-HFBR-1414
  - Optical Output

Open Source Event Receiver - Introduction

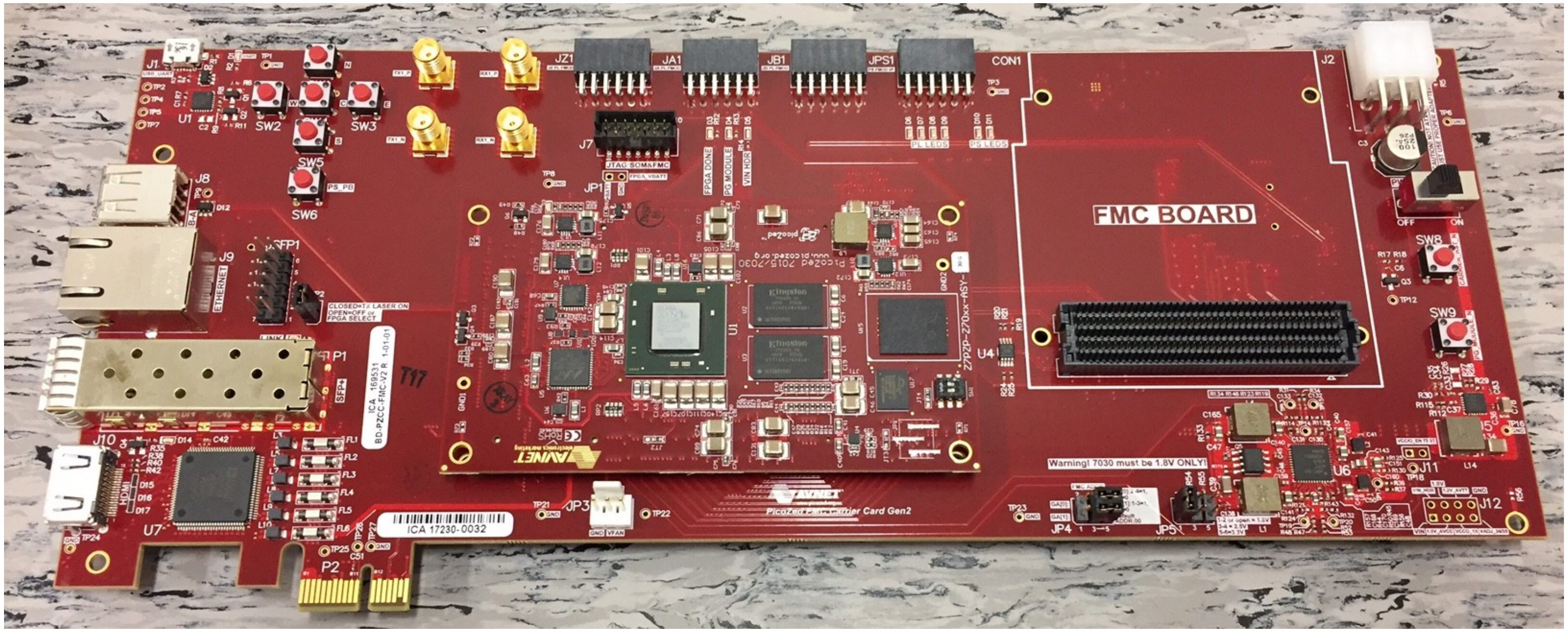
- What is the Open Source Event Receiver
  - Basic building block required to build devices receiving the MRF Timing protocol including but not limited to Delay Compensation capability
- What is it not
  - it is not a replacement firmware for current MRF products
  - it is not a complete Event Receiver with MRF product compatible register map
    - No bus/register interface
    - No pulse generators
    - etc.
- Available on GitHub https://github.com/jpietari/mrf-openevr

Open Source Event Receiver - Requirements

- Hardware
  - Xilinx Kintex-7 based FPGA with GTX transceivers **ONLY!**
  - Zynq 7Z030 is Kintex-7 based
  - SFP Transceiver
  - Reference clock for GTX
  - Example design built for
    - Avnet PicoZed 7Z030
    - Avnet PicoZed FMC Carrier Card V2
- Software
  - Xilinx Vivado 2017.4 (Free WebPack version is sufficient)
- Xilinx programming cable
  - e.g. Platform Cable USB II

Avnet PicoZed FMC carrier with 7Z030 SOM

- Avnet PicoZed AES-Z7PZ-7Z030-SOM-G
- Avnet PicoZed FMC carrier AES-PZCC-FMC-V2-G
  - Zynq 7Z030 incorporates
    - Kintex-based FPGA core
    - Four GTX transceivers
    - Dual-core ARM Cortex-A9
- This kit has everything from the hardware point of view to be used as an event receiver


FMC BOARD
SFP1
ETHERNET
HDMI
Warning! 7030 must be 1.8V ONLY!
PicoZed FMC Carrier Card Gen2