Monitoring and control. DAQ and FEE *Introduction*

Aleksandr Boikov

E-mail: Boikov@jinr.ru

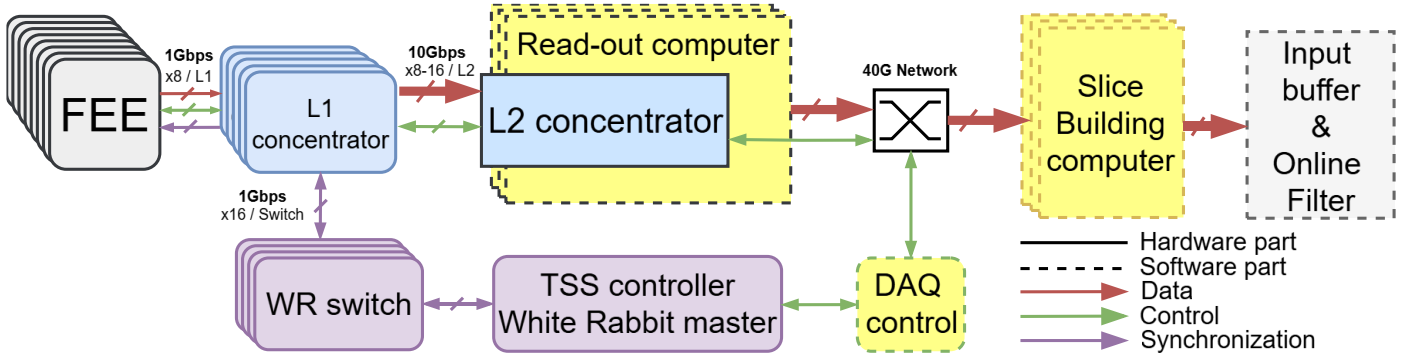
21.11.25

DAQ Struct

Elements of DAQ:

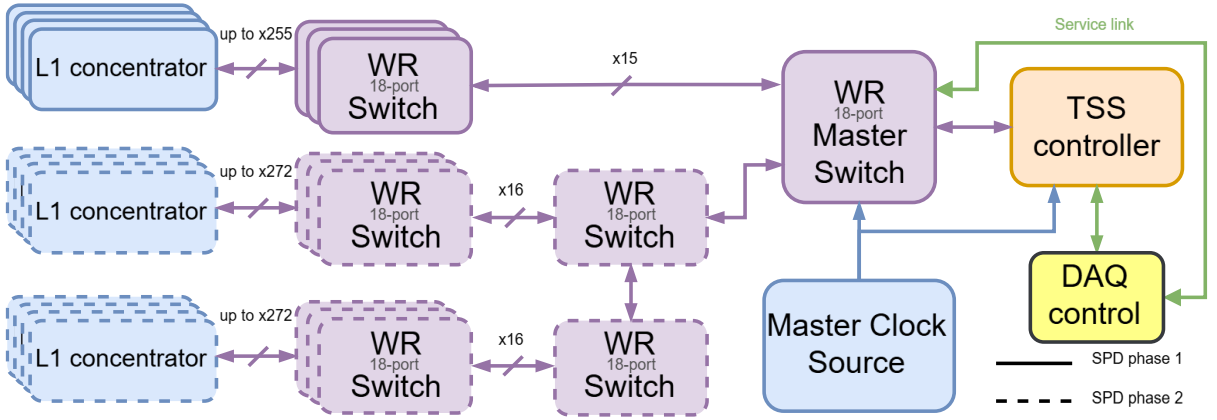
- Read-out chain
  - L1 – Level 1 concentrator
  - L2 – Level 2 concentrator
  - Read-out computer
- Time Synchronization System
- Slice building system

SPD Stage	FEE Outputs	L1 (8 ports)	L2 (16 ports)	Read-out PC	Slice Builders
First	~1292	>165	>12	>6-12	>5
Second	~5800	>730	>50	>25-50	>40



TSS Components:

- TSS controller
- Ultra-stable clock source
- 18-port White Rabbit-enabled switches
- White Rabbit nodes integrated into L1 hubs



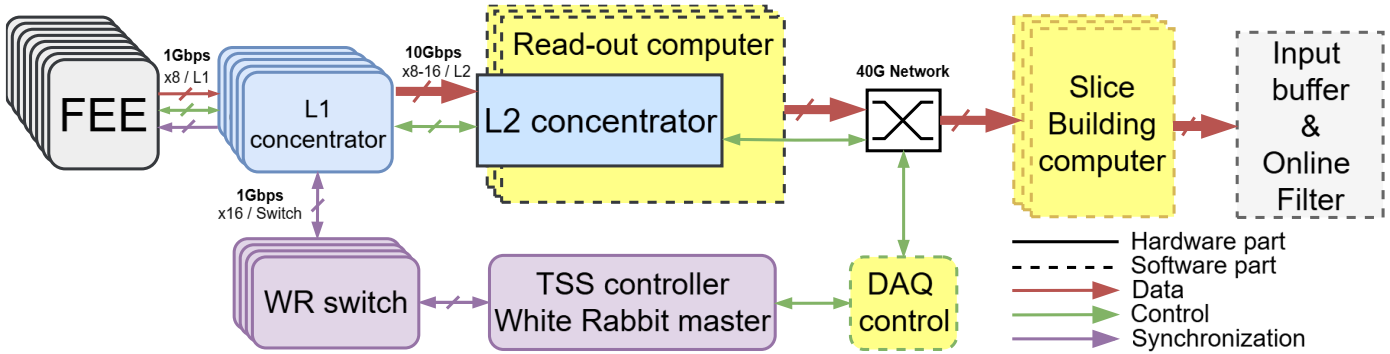
Monitoring and control

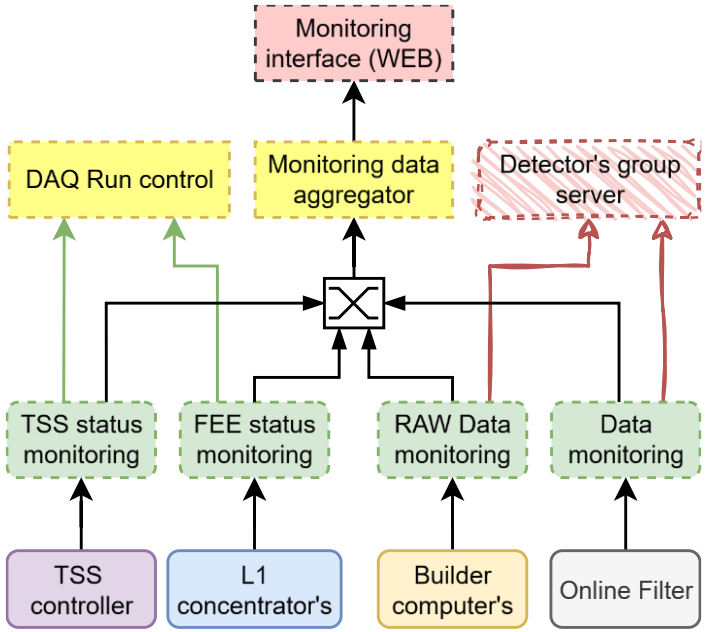
Base logic:

- The entire system is controlled via “DAQ Control”.
- All FEE management is performed via a dedicated read chain.
- Each L1 concentrator in this network will be visible as a separate device.

Monitoring:

- FEE status and data volume from each FEE
- Data transmission errors
- Synchronization system status
- Raw data provision for detector groups
- Online filtered data provision for detector groups





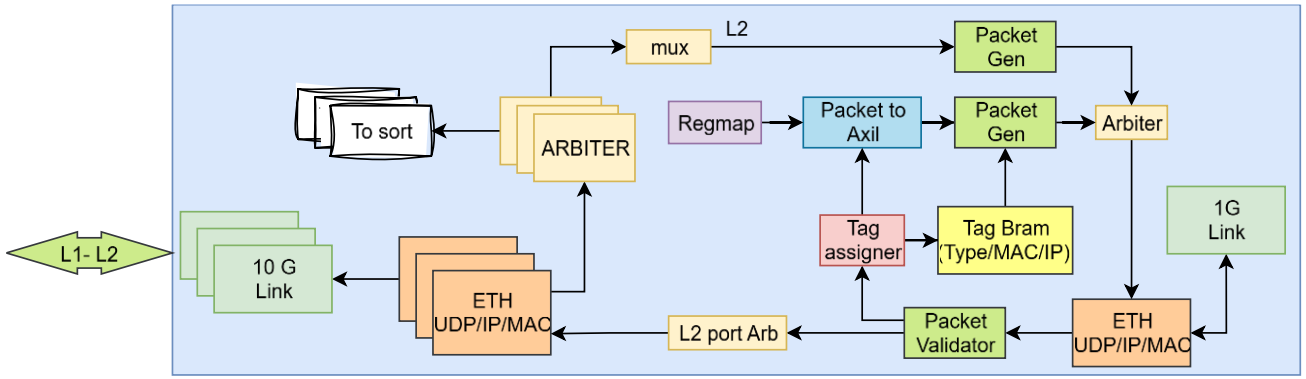
Read-out chain & Axil to Packet

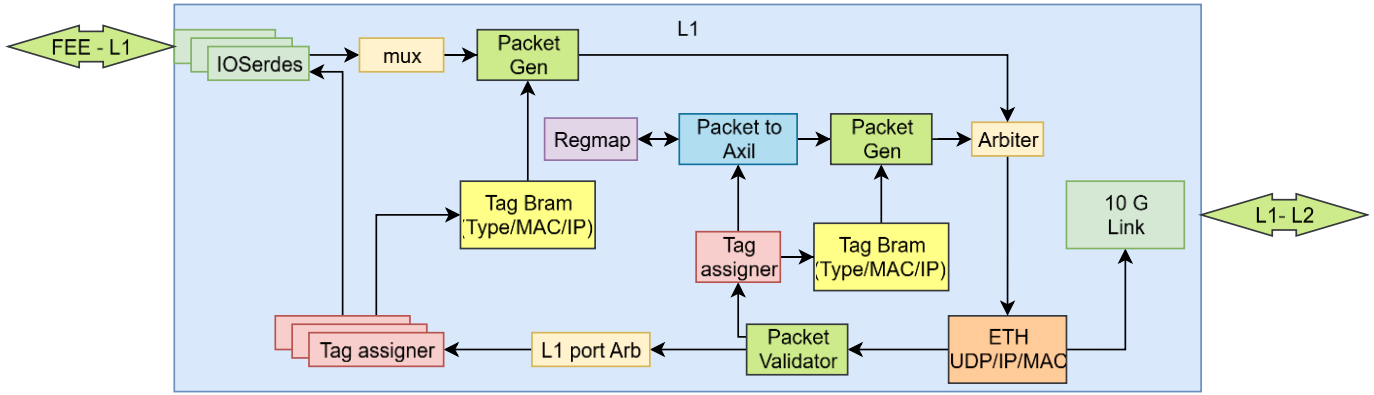
Control and monitoring packets are transmitted via L2 to L1 and the FEE.

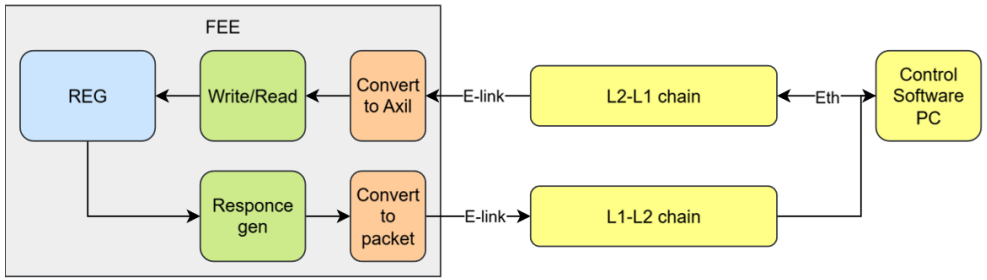
The "Packet Validator" module determines what to do with the received data. If the address matches the device, the packet is passed to the internal register management module. Otherwise, the packet is forwarded to the next hub level.

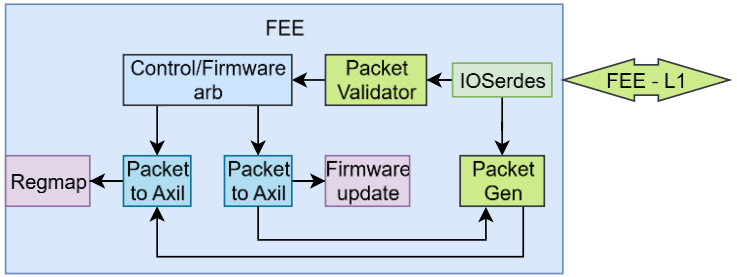
The packet is transmitted to the FEE provided that the corresponding board with the correct identifier is connected to the L1 port.

For L1 and L2 control, UDP is used with AXIL bus encapsulation in the payload. For the FEE, an additional addressing header and AXIL bus encapsulation are assumed.

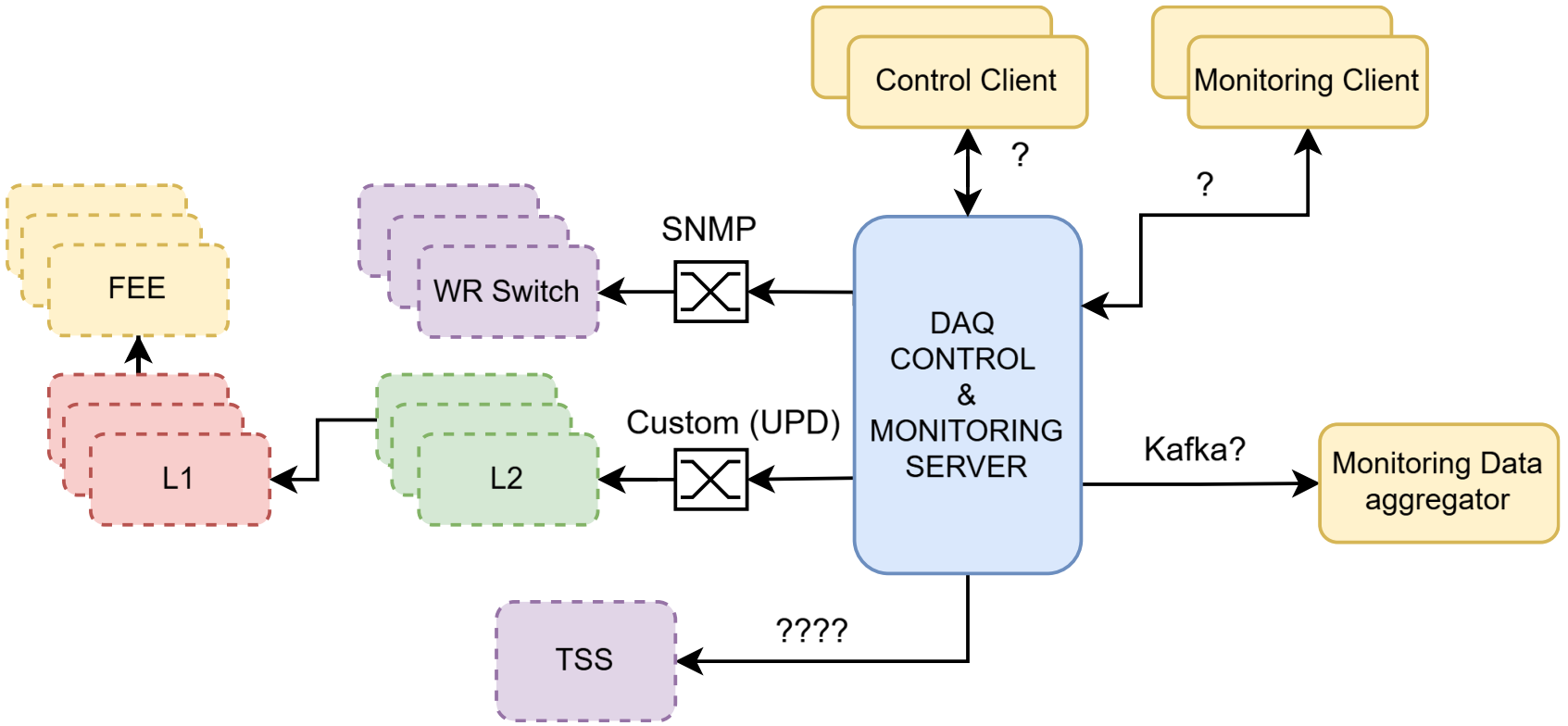








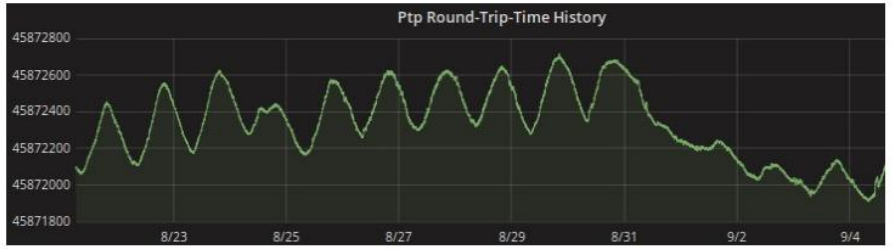
Hardware monitoring scheme

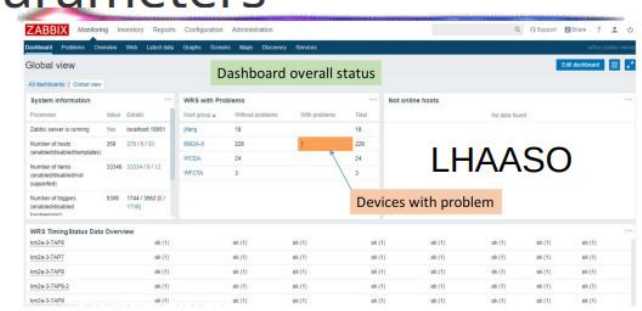


https://events.geant.org/event/1207/contributions/1179/attachments/779/1115/Geant_WR_management_Adam_Wujek.pdf

Network monitoring (SNMP)

- Timing monitoring (Sync status)
- WRS: more than 300 specific parameters
- WRPC: ~70 parameters





Monitor: WRS sync status, packet flow of WRS port, RTT, temperature, etc..

Source: https://ohwr.org/project/white-rabbit/wikis/uploads/ea8d8eac37a97df1cf3cb097d081b8f4/WR_at_LHAASO.pdf

WRPC: SNMP

- WR-WRPC-MIB (in wrpc-sw repo)
  - No status OIDs
  - Port's statistics
  - PTP/WR timing status and configuration
  - SFP calibration database
  - SFP monitoring (temperature, RX/TX power)
- SNMP can be used to configure some parameters:
  - SFP database
  - Init script
  - Remote shell command execution
- Not Secure! No SNMP v3

WRPC: other supported protocols

- VLANs (limit access between network parts)
- Syslog (logging), events like:
  - boot up
  - link down/up
  - sync lost
  - sync recovered
  - Temperature over threshold
- BOOTP
- Netconsole

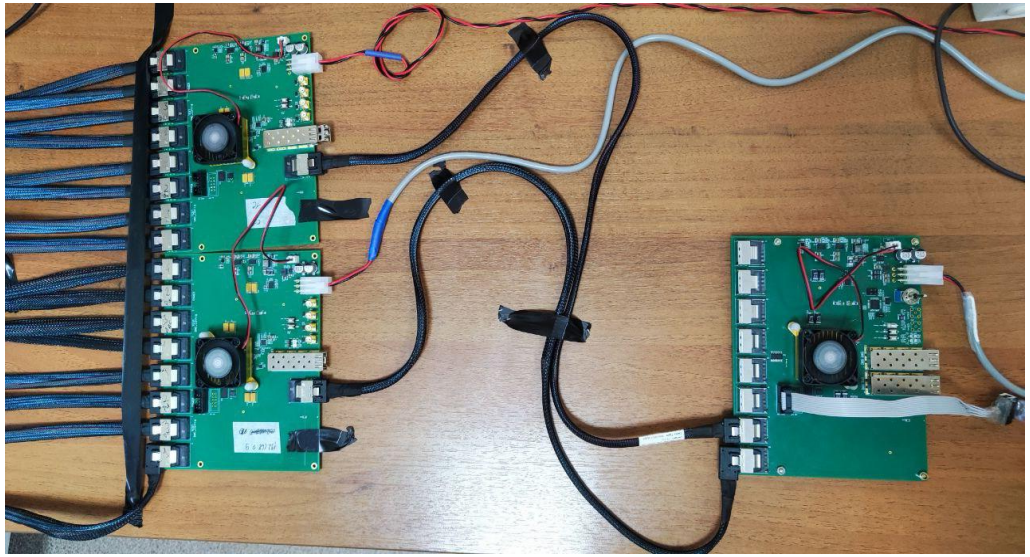
Thank you for your attention

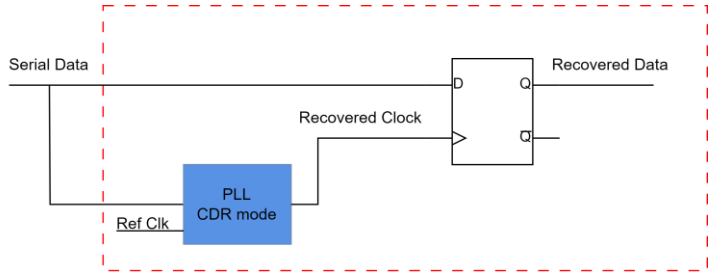
E-mail: Boikov@jinr.ru

E-Link

Dedicated lines:

- Start of slice
- Start of frame
- Set Next Frame
- TSS global clock
- Data line for FEE-L1 (Max 1250 Mbps) & ~~Clock~~ (Data and feedback)
- Data line for L1-FEE (Max 1250 Mbps) & Clock (Control and firmware update)
- 8 lines for slow interfaces, handshakes, etc.

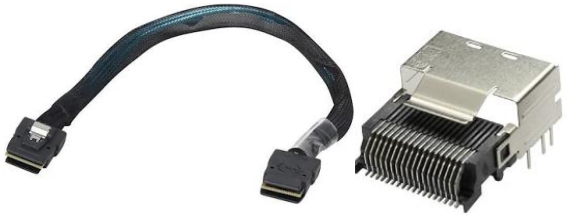




Bandwidth:

150 Mbps – OK
500 Mbps – OK
750 Mbps – OK
1000 Mbps – OK
1250 Mbps – OK

MiniSAS



Pin function	Side B Pin	Side A Pin	Pin function
GND	B1	A1	GND
Tx0+	B2	A2	Rx0+
Tx0-	B3	A3	Rx0-
GND	B4	A4	GND
Tx1+	B5	A5	Rx1+
Tx1-	B6	A6	Rx1-
GND	B7	A7	GND
Unipolar 0	B8	A8	Unipolar 7
Unipolar 1	B9	A9	Unipolar 3
Unipolar 2	B10	A10	Unipolar 4
Unipolar 6	B11	A11	Unipolar 5
GND	B12	A12	GND
Tx2+	B13	A13	Rx2+
Tx2-	B14	A14	Rx2-
GND	B15	A15	GND
Tx3+	B16	A16	Rx3+
Tx3-	B17	A17	Rx3-
GND	B18	A18	GND

RegMap (SystemRDL)

SYSTEMS INITIATIVE

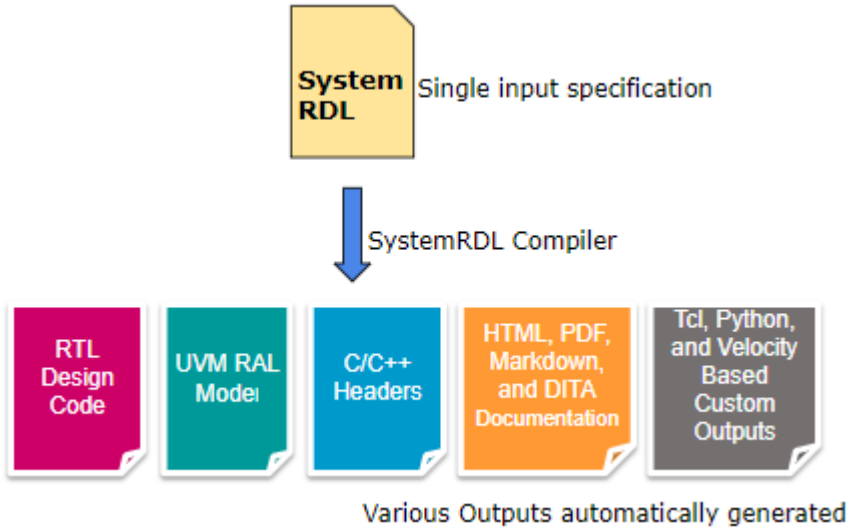
SystemRDL 2.0
Register Description Language

PeakRDL

This tool can:

+

- Process SystemRDL 2.0 register descriptions.
- Generate synthesizable SystemVerilog RTL register blocks
- Generate a C register abstraction header for software.
- Import & export IP-XACT XML.
- Create rich and dynamic HTML documentation.
- Build a UVM register model abstraction layer.



System RDL language

```
`ifdef OFFSET
    `define OFFSET_CONDITION OFFSET_GLOBAL::teng_map_addr
`else
    `define OFFSET_CONDITION 0
`endif
addrmap teng_map {
    name = "LINK CONFIG REG";
    desc = "Register description of LINK MAIN MODULE. Includes setti

    default regwidth = 32;
    default addressing = fullalign;
    default accesswidth = 32;
    default sw = rw;
    default hw = r;

    reg {
        name = "source_mac_part1";
        regwidth = 32;
        field {
            sw = w;
            hw = r;
            desc = "Source mac addr1";
        } source_mac_part1 [32] = 32'h00_1b_21_ed;
    } source_mac_part1 @ `OFFSET_CONDITION;

    reg {
        name = "source_mac_part2";
        regwidth = 32;
        field {
           sw = w;
           hw = r;
           desc = "Source mac addr2";
        } source_mac_part2 [16] = 16'h00_FE;
        field {
        } reserved [16];
    } source_mac_part2;

    reg {
        name = "source_ip";
        regwidth = 32;
        field {
            desc = "Source ip addr";
        } source_ip [31:0] = 32'hC0_A8_00_FE;
    } source_ip;
```

Docs (markdown)

teng_map address map

- Absolute Address: 0x0
- Base Offset: 0x0
- Size: 0x134

Register description of LINK MAIN MODULE. Includes settings

Offset	Identifier	Name
0x100	source_mac_part1	source_mac_part1
0x104	source_mac_part2	source_mac_part2
0x108	source_ip	source_ip
0x10C	dest_mac_part1	dest_mac_part1
0x110	dest_mac_part2	dest_mac_part2
0x114	dest_ip	dest_ip
0x118	config_ports	Config port
0x11C	jitter_ports	Jitter port
0x120	data_ports	Data port
0x124	gateway_ip	gateway_ip
0x128	subnet_mask	subnet_mask
0x12C	padding	padding
0x130	configs	config

System Verilog module

```
module teng_map (
        input wire clk,
        input wire rst_n,

        output logic s_axil_awready,
        input wire s_axil_awvalid,
        input wire [31:0] s_axil_awaddr,
        input wire [2:0] s_axil_awprot,
        output logic s_axil_wready,
        input wire s_axil_wvalid,
        input wire [31:0] s_axil_wdata,
        input wire [3:0]s_axil_wstrb,
        input wire s_axil_bready,
        output logic s_axil_bvalid,
        output logic [1:0] s_axil_bresp,
        output logic s_axil_arready,
        input wire s_axil_arvalid,
        input wire [31:0] s_axil_araddr,
        input wire [2:0] s_axil_arprot,
        input wire s_axil_rready,
        output logic s_axil_rvalid,
        output logic [31:0] s_axil_rdata,
        output logic [1:0] s_axil_rresp,

        output teng_map_pkg::teng_map__out_t hwif_out
    );
```

C-h file for software

```
// Generated by PeakRDL-cheader - A free and open-source header generator
//  https://github.com/SystemRDL/PeakRDL-cheader

#ifndef REG_LINK_H
#define REG_LINK_H

#ifdef __cplusplus
extern "C" {
#endif

#include <stdint.h>
#include <assert.h>

// Reg - teng_map::source_mac_part1
#define TENG_MAP__SOURCE_MAC_PART1__SOURCE_MAC_PART1_bm 0xffffffff
#define TENG_MAP__SOURCE_MAC_PART1__SOURCE_MAC_PART1_bp 0
#define TENG_MAP__SOURCE_MAC_PART1__SOURCE_MAC_PART1_bw 32
#define TENG_MAP__SOURCE_MAC_PART1__SOURCE_MAC_PART1_reset 0x1b21ed
typedef union {
    struct __attribute__ ((__packed__)) {
        uint32_t source_mac_part1 :32;
    } f;
    uint32_t w;
} teng_map__source_mac_part1_t;

// Reg - teng_map::source_mac_part2
#define TENG_MAP__SOURCE_MAC_PART2__SOURCE_MAC_PART2_bm 0xffff
#define TENG_MAP__SOURCE_MAC_PART2__SOURCE_MAC_PART2_bp 0
#define TENG_MAP__SOURCE_MAC_PART2__SOURCE_MAC_PART2_bw 16
#define TENG_MAP__SOURCE_MAC_PART2__SOURCE_MAC_PART2_reset 0xfe
#define TENG_MAP__SOURCE_MAC_PART2__RESERVED_bm 0xffff0000
#define TENG_MAP__SOURCE_MAC_PART2__RESERVED_bp 16
#define TENG_MAP__SOURCE_MAC_PART2__RESERVED_bw 16
typedef union {
    struct __attribute__ ((__packed__)) {
        uint32_t source_mac_part2 :16;
        uint32_t reserved :16;
    } f;
    uint32_t w;
} teng_map__source_mac_part2_t;
```