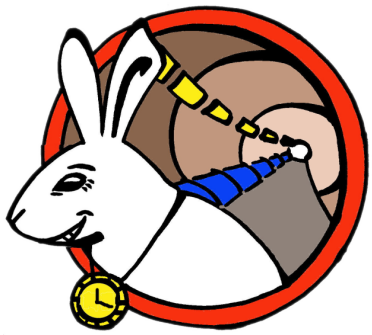
European Organization for Nuclear Research
Organisation Europeenne pour la Recherche Nucleaire



# White Rabbit switch

## Technical specification

Version: **20090227**

**Abstract**

This document describes design guidelines and technical details for White Rabbit switch microTCA MCH and AMC modules. The switch is core component of White Rabbit network - synchronous Ethernet-based new timing and control network at CERN.

Tomasz Włostowski
CERN AB-CO-HT
Tel : +41 (0)22 76 79262
Bat : 864-1-A07

Geneve, February 27, 2009

# Contents

# 1 Background

The White Rabbit is next-generation deterministic network based on synchronous Ethernet, allowing for low-latency deterministic packet routing and transparent, high precision timing transmission. The network consists of master node (only one active at a time) which provides time and frequency reference, switches interconnected through fibers or twisted-pair copper in star topology and slave nodes.

# 2 Requirements

Below are listed requirements for the switch design, ordered by importance:

1. **high performance timing transmission** – the switch should be able to replicate the master clock received from uplink port with maximum skew of 200 ps
2. **deterministic routing** – for certain class of packets, packet processing delay introduced by the switch should be constant or at least less than certain value (64 x 8 ns)
3. **compatibility** with existing standards: Ethernet, PTP (IEEE1588)
4. **fault tolerance** for critical control traffic and timing
5. **interoperability** with hardware/software not compliant with White Rabbit standard (with limited features)
6. **modular design** providing scalability by adding modules with additional ports
7. **remote management** via SNMP and/or remote console
8. **local timing receiver** mode - switch MCH working as timing receiver for cards in the same crate and providing precise timing to each card without using PTP and deterministic Ethernet.
9. capability of working in different topologies than star (e.g. token ring for beam orbit applications)
10. ultra-low latency "hub" mode with no collision handling (first-come, first-go)
11. security/node authentication

# 3 Block diagram

The complete White Rabbit switch consists of two different modules (described in following chapters) which are working in microTCA crate with appropriate backplane:

- WR MCH module with two uplink ports acting as master card for WR AMCs and responsible for high-level switch features like switch/crate management, PTP, etc. The MCH can also work as standalone GbE switch and easy to use (no PTP) timing receiver for any cards residing in the same crate.
- WR AMC cards containing "subswitches" with 8 downlink ports each uplinked through backplane to WR MCH.

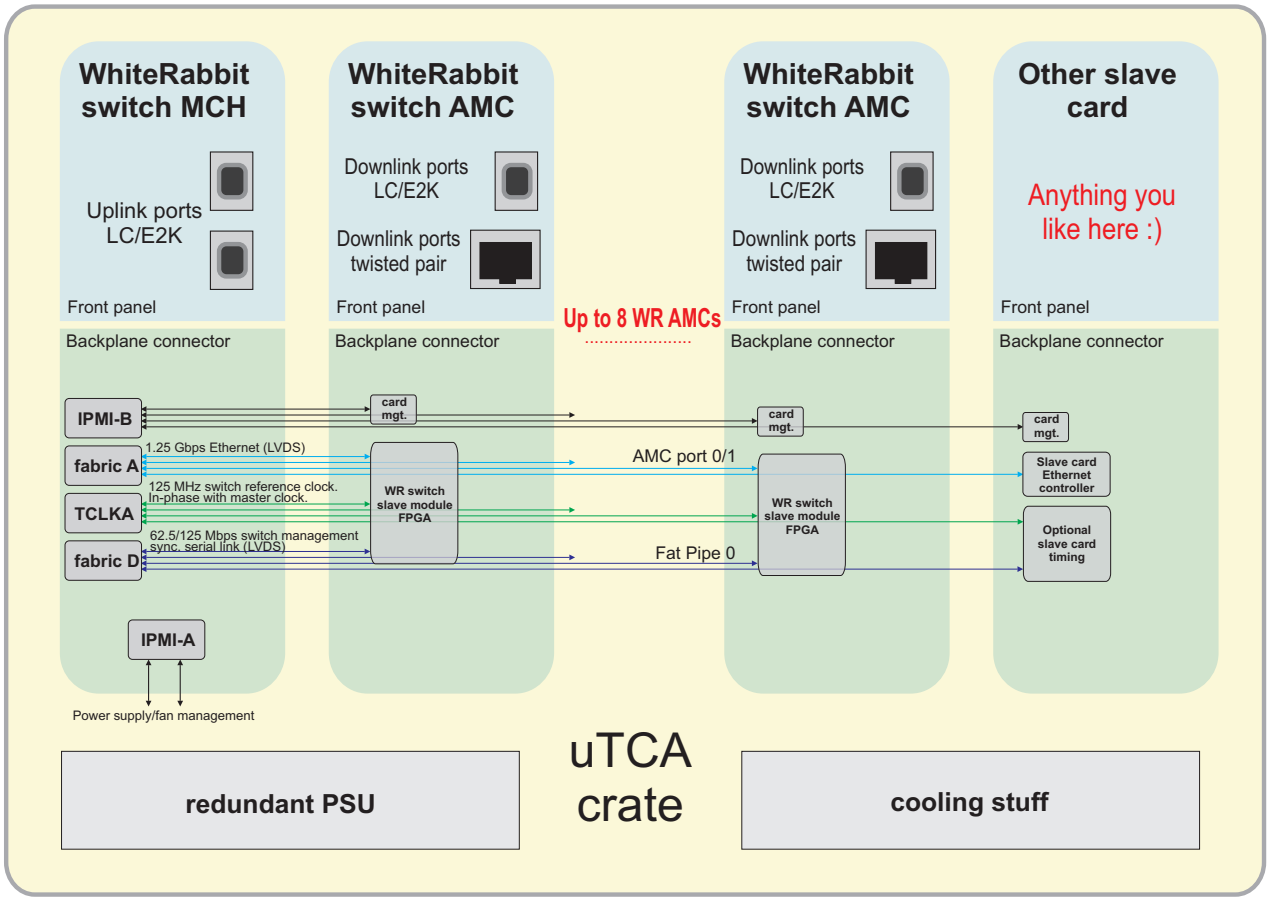



Figure 1: General block diagram of WhiteRabbit switch

General block diagram, showing how these modules are connected in the crate is depicted on figure 1. System uses following uTCA backplane lanes:

- Fabric A (from WR MCH to AMC slots port 0/1) - main gigabit Ethernet link (LVPECL), interconnecting WR MCH with subswitch modules in AMCs or other cards supporting GbE.
- Fabric D (from WR MCH to AMC slots port 4 - fat pipe 1) - additional synchronous serial link (LVPECL/LVDS) running at 125 Mbps. Used for management and configuration of WR AMCs and for transmitting data between WR MCH and WR AMCs which

cannot be multiplexed into main Ethernet link without making it nondeterministic (like timestamps). It is also used to transmit PPS/timecodes to slave cards with simplified timing receivers (not requiring PTP).

- Telco clock A (from WR MCH TCLKA to AMC port CLKA) - 125 MHz low-jitter reference clock phase-locked with System Timing Master clock. These clocks are generated independently for each card in the crate.

# 4 WR MCH Hardware

In this section, WR MCH hardware is described.

## 4.1 Inputs and outputs

Front panel I/O:

- **Two SFP module sockets** for uplink ports (UP0, UP1). Uplink port 0 is the primary uplink port, used in normal conditions. Uplink port 1 is used as timing and HP source when primary uplink connection is not functioning properly.
- **125 MHz reference clock input** (EXTREFIN˙125M). Used to provide external 125 MHz clock as a reference for downlink ports instead of clock recovered from uplinks. Can be used for daisy-chaining multiple MCHs.
- **10 MHz reference clock input** (EXTREFIN˙10M). Allows for using 10 MHz reference clock from cesium/GPS.
- **125 MHz reference clock output** (EXTREFOUT) with 125 MHz network reference clock, recovered from uplink or provided by external input. Used for daisy-chaining MCHs.
- **PPS input** (EXTPPSIN). Receives PPS pulses from external source. Along with EXTREFIN can be used to synchronize the switch to external timing source.
- **PPS output** (EXTPPSOUT). Outputs PPS signal. For daisy-chaining crates or synchronizing external devices.
- **Configurable RS232 port** (RS232). RS232 port, which can be used either by local serial management console or as input of NMEA timecode from GPS/cesium.
- **100Mbit Ethernet port** (ETH). Twisted-pair ethernet port connected to management CPU. Allows for using external, non-WR network to manage WR switches and uTCA crates.
- **LEDs**. 8 dual-color LEDs for debugging/informational purposes.

microTCA Backplane I/O:

- 8 LVPECL Gigabit Ethernet lanes (from WR MCH to each AMC slot port 0/1)
- 8 LVDS management i/f lanes (from WR MCH to each AMC slot port 4)
- 8 LVDS 125 MHz TCLKs (from WR MCH to each AMC slot port CLKA)
- IPMB-A/B to PSU and fans, IPMB-L lines to first 8 AMC slots.
- power and auxillary uTCA control signals

## 4.2 Uplinks and clock recovery section

The aim of this block is to generate in-phase reference clock from data stream coming to uplink ports. It includes compensation for phase shifts introduced by link and PLL cleaning. The data stream itself is passed unmodified to main WR MCH FPGA.

Block diagram of clocking system is depicted on figure 2. For clarity, 2nd uplink port PHY is not shown on the picture.

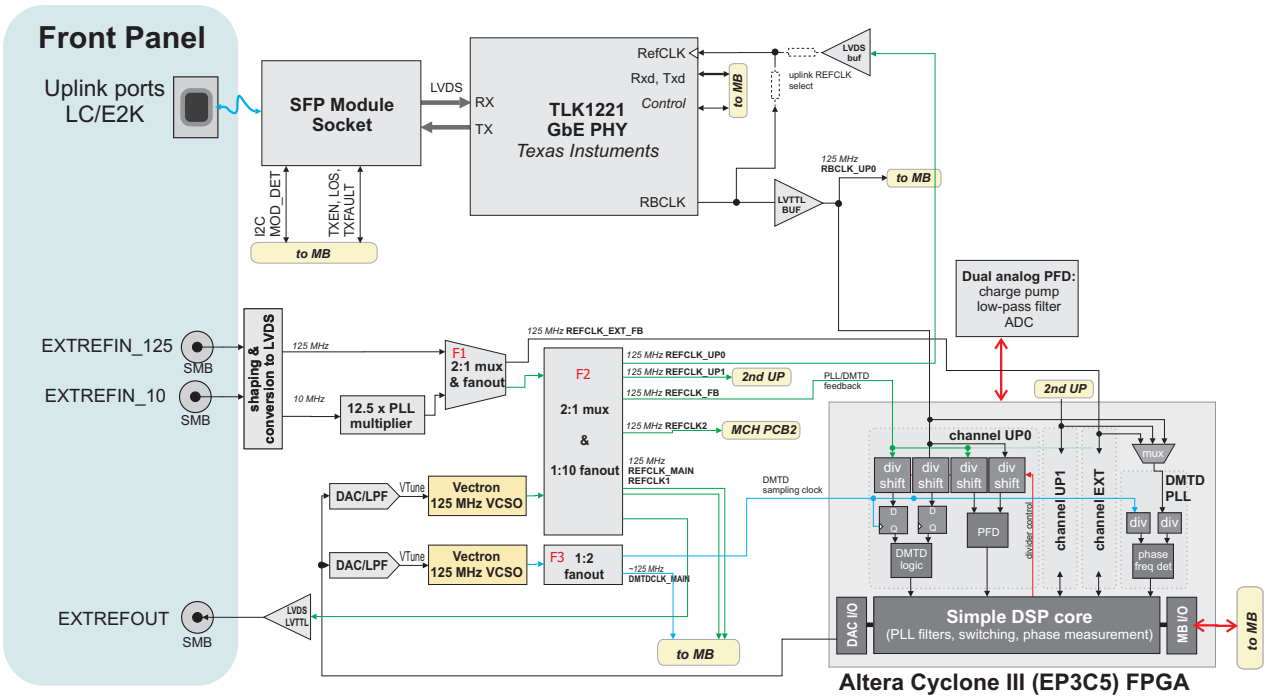



Figure 2: Block diagram of WR MCH clock recovery/uplink section

We can get uncompensated reference clock from three sources: uplink 0/1 PHYs (UP0, UP1) and external clock inputs (EXTREFIN˙10M and EXTREFIN˙125M).

Clock embedded into data stream is received from fiber uplink by SFP module and passed to Texas Instruments TLK1221 deterministic GbE PHY. TX/RX data is routed directly from/to main WR MCH FPGA. Recovered symbol clock (RBCLK) is passed to:

- DMTD/PLL feedback in clocking FPGA
- main WR MCH FPGA (for packet timestamping)
- (optional) as transmit clock for uplink PHY (instead of REFCLK˙UPx)

External reference clocks can be supplied to EXTREFIN connectors. Device supports both 125 MHz native Ethernet frequency and 10 MHz GPS/cesium frequency. EXTREFIN inputs support variety of signal standards (TTL/CMOS/ECL/clipped sine) by using Bruce Griffith's clock shaping input circuit. 10 MHz input is multiplied 12.5 times by additional PLL (Onsemi NB3N3020). Both external clocks are then passed to LVDS mux/fanout F1. First F1 output is connected directly to main REFCLK mux/fanout, bypassing PLLs. Another output delivers feedback external clock to clocking FPGA (in case we wanted to phase-shift or clean it).

Clock recovery is performed by small Altera Cyclone3-series FPGA and two external VCSOs (Voltage Controlled SAW Oscillators) driven by DACs:

- upper VCSO is the main reference clock (REFCLK) generator,
- lower VCSO produces DMTD sampling clock (DMTDCLK). DMTD clock frequency can be programmed via software depending on oscillator properties and required measurement accuracy/speed.

The FPGA contains three switchable PLL/phase shifter channels and custom DSP core which performs all necessary calculations, filtering and driving VCSO tuning DACs. Each channel consists of:

- digital DMTD (a.k.a "Pablo's DMTD") phase detector - the preferred method, explained in detail in appendix 1.
- standard phase-frequency detector with appropriate dividers. Phase comparison result can be obtained by uncorrelated sampling technique or by using external analog charge pump, loop filter and ADC.

We can choose between both detectors or combine them together. Each channel measures phase shift and detects frequency difference between REFCLK and input clocks from UP0, UP1 and EXTREFIN. Knowing the phase difference, we can compensate it, and thus recover in-phase master clock.

Compensated REFCLK is split into 7 separate lanes by 2:10 LVDS fanout F2. Outputs of fanout are assigned as following:

- REFCLK˙MAIN, REFCLK1 - reference clocks for MCH main board.
- REFCLK˙FB - feedback clock for phase detectors in clocking FPGA
- REFCLK˙UP0, REFCLK˙UP1 - optional transmit clocks for uplink PHYs
- REFCLK2 - clock source for uTCA TCLKA fanout (on MCH PCB 2)
- EXTREFOUT - external clock output (through LVDS-LVTTL buffer)

When none of reference clock sources is functioning, oscillator operates in free-running mode, still providing proper clock for FPGAs (although not locked to any reference).

For flexible firmware update, clocking FPGA is booted by MCH CPU via JTAG or passive serial bus. Communication between clocking FPGA and main FPGA is done using CMI (Clock Mezzanine Interface) link, consisting of SPI bus and few user-programmable digital lines.

## 4.3 Main digital section

This is the main part of WR MCH, containing the big FPGA which takes care of packet routing and CPU which handles switch/microTCA crate management and PTP tasks. The block diagram is shown on figure 3

### 4.3.1 Main FPGA

We've chosen Altera's biggest Cyclone-3 device (EP3C120) in 780-pin BGA package as it has the biggest total size of builtin memory blocks in this market segment which is crucial for switch application. Inside the FPGA there are:

- WR switch core
- Local Ethernet controller, interfacing the WR network with switch management CPU

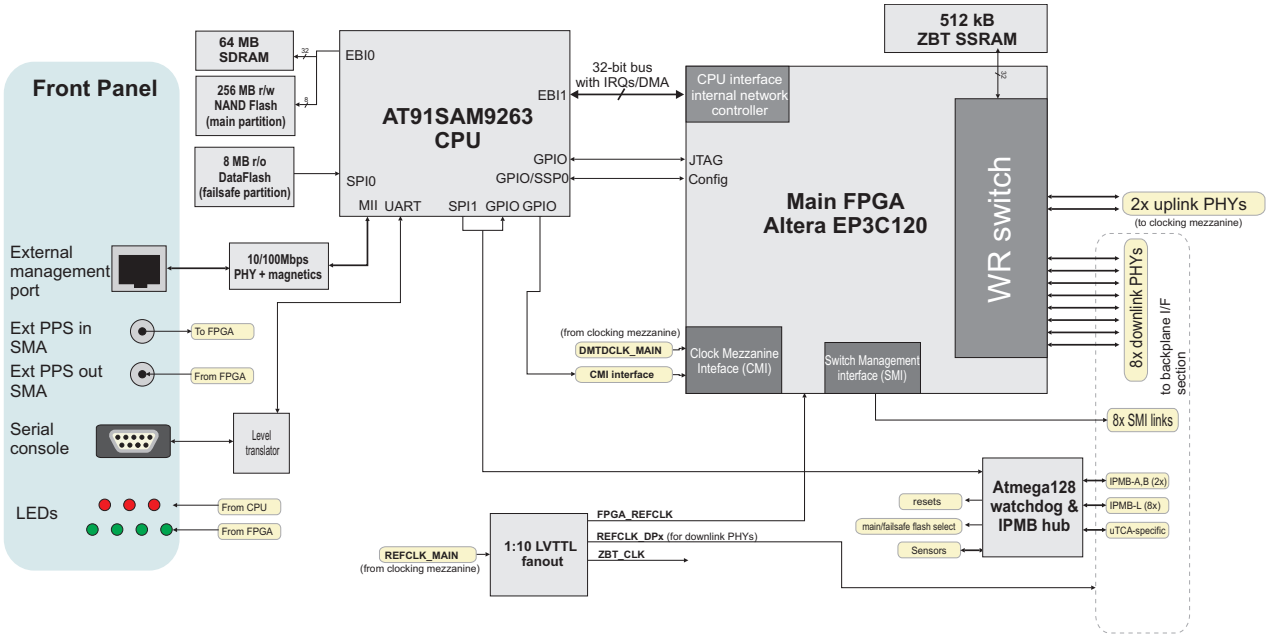



Figure 3: Block diagram of WR MCH main digital section

- CPU interface - EBI slave for accessing various configuration registers. It uses 32-bit asynchronous bus with wait states, additional interrupt lines and DMA request signal.
- Clock management interface for communicating with clocking mezzanine board
- Switch management interface endpoints
- Some GPIO ports for driving LEDs and SFP detection

FPGA is directly interfaced via 32-bit bus with 512 Kbyte ZBT synchronous SRAM memory, used to store routing tables which need fast access. 2 uplink and 8 downlink PHYs with RX clocks and appropriate control signals are also directly coupled with FPGA.

REFCLK for FPGA, ZBT mem and downlink PHYs is provided by 1:10 LVTTL fanout. We have DMTDCLK supplied from clocking mezzanine for downlink port phase measurement.

CPU runs at its own clock, which can be also used for uncorrelated sampling phase detector on clocking mezzanine.

### 4.3.2 CPU

For the CPU we have decided to choose Atmel AT91SAM9263 chip (ARM926E core running at 200 MHz) because of its popularity, well-known architecture (ARM) and OS support (Linux). Following CPU peripherals are used:

- External bus interface 0 (EBI0) connected to 64 MBytes of SDRAM (two 16-bit bus chips) and 256 MB NAND flash chip containing main flash firmware
- External bus interface 1 (EBI1) operating in Static Memory mode, connected to CPU i/f in main FPGA
- SPI interface (SPI) connected to DataFlash memory containing failsafe boot image
- Ethernet MAC (EMAC) connected via appropriate magnetics to 100Base-T RJ45 port on front panel of MCH (local ethernet management)
- UART routed to connector on front MCH panel (local serial port)

- GPIO pins connected to FPGA configuration pins/JTAG port, used to boot the FPGA.
- SPI interface and GPIO pins connected to watchdog AVR microcontroller

CPU is interfaced with FPGA using 32-bit asynchronous bus (with external wait state support) and few independent interrupt lines.

Main tasks of CPU are:

- PTPv2 daemon
- WRP protocol deamon for WR node discovery and sub-nanosecond phase sync
- Switch management through SNMP/remote console.
- Handling of high-layer IEEE802.1x protocols (multicasting, spanning tree, GMRP/-GARP)
- Configuration of special WR features

### 4.3.3 Watchdog MCU

In order to achieve high system reliability and fault tolerance, there is another, small Atmega128 microcontroller implementing following functions:

- Watchdog unit, periodically polling important switch components (FPGAs, CPU), checking if they are alive and resetting them if needed
- IPMB bus I2C hub
- Power supervisor for the card - powering the card, initializing low-level uTCA stuff, handling hot-swap
- Performing failsafe boot sequence after faulty firmware update
- Monitoring temperature sensors

Watchdog MCU is interfaced with main CPU via SPI link and external interrupt line. Also it has I/O lines going to both FPGAs which are used by polling mechanism. The MCU is supplied from 3.3V Management Power line.

## 4.4 Backplane interface

The backplane interface is depicted on figure 4. Main components are listed below:

- 8 TLK1221 PHYs driving main GbE links
- 8 LVDS complementary buffers for driving SMI links
- 8-output LVDS clock fanout driving the TCLKA fabric lines

Apart from GbE link, management link and clocks there are several MCH-specific signals (such as IPMB busses to each AMC slot and fan controller/PSU).

## 4.5 Power supply

As the microTCA backplane provides only +12 V supply voltage, separate power supply is required. Its block diagram can be found on figure 5.

Main +3.3 V/2.5 V digital power supplies are done in buck topology, with external discrete MOSFET switches, allowing max. 4 A of output current. +1.2 V core supply for main FPGA

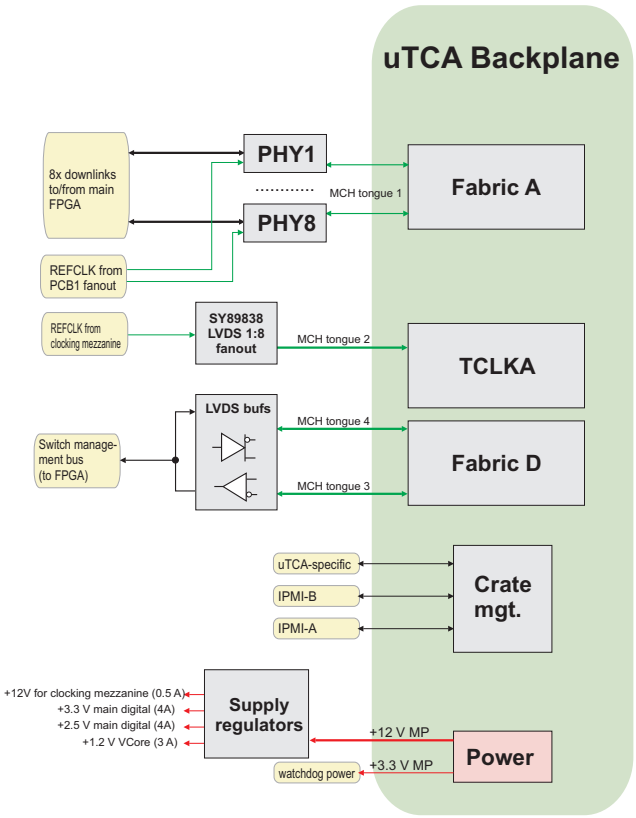



Figure 4: Block diagram of WR MCH backplane interface

is produced by integrated switched regulator. All other supplies come from standard LDO regulators.

Supply voltage for VCXOs comes from separate LT1762 ultra-low noise LDO to minimize impact of supply voltage noise on REFCLK performance.

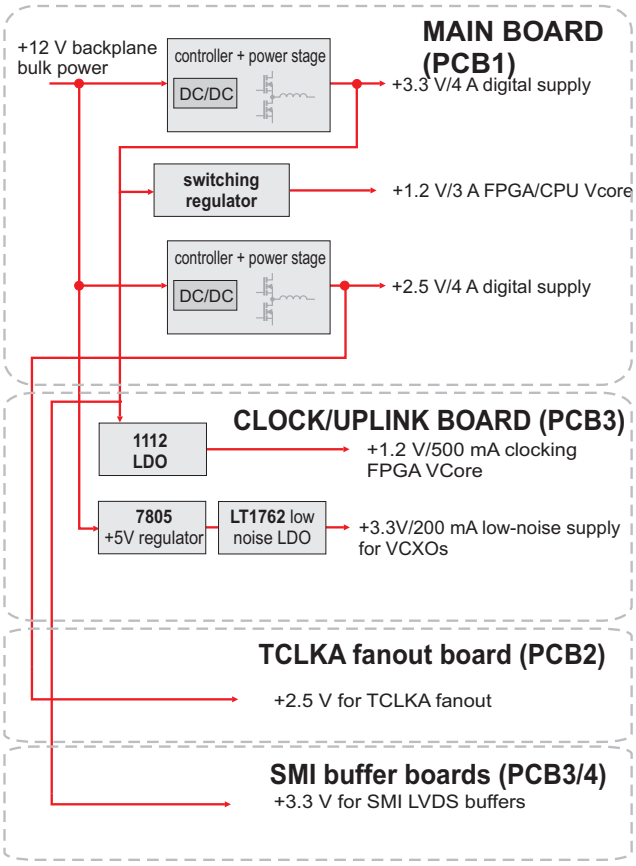



Figure 5: Block diagram of WR MCH power supply

# 5 WR MCH implementation

This section describes important implementation details of MCH module. It also explains some ideas used in design.

## 5.1 PCB and mechanical considerations

The switch design is too complex to be fit on single uTCA card. Also the required backplane layout forces us to use separate PCBs. Proposed switch physical implementation is shown on figure 6.

PCBs are interconnected with dense (0.8mm) SMD connectors. For mechanical stability, there are spacers binding boards to each other (4 spacers between adjacent boards). In order to preserve board space and minimize the cost, we've decided to use PCB edge connectors for backplane interface instead of special (and very expensive) uTCA snap-on connectors.

Detailed block distribution on PCBs is depicted on figure 6. General idea is to separate clocking system (which resides on PCB3) from generic digital stuff which is not very likely to be changed (PCB1).

Another problem we have faced is very limited space on uTCA card front panel. Figure 7 shows proposed front panel layout.

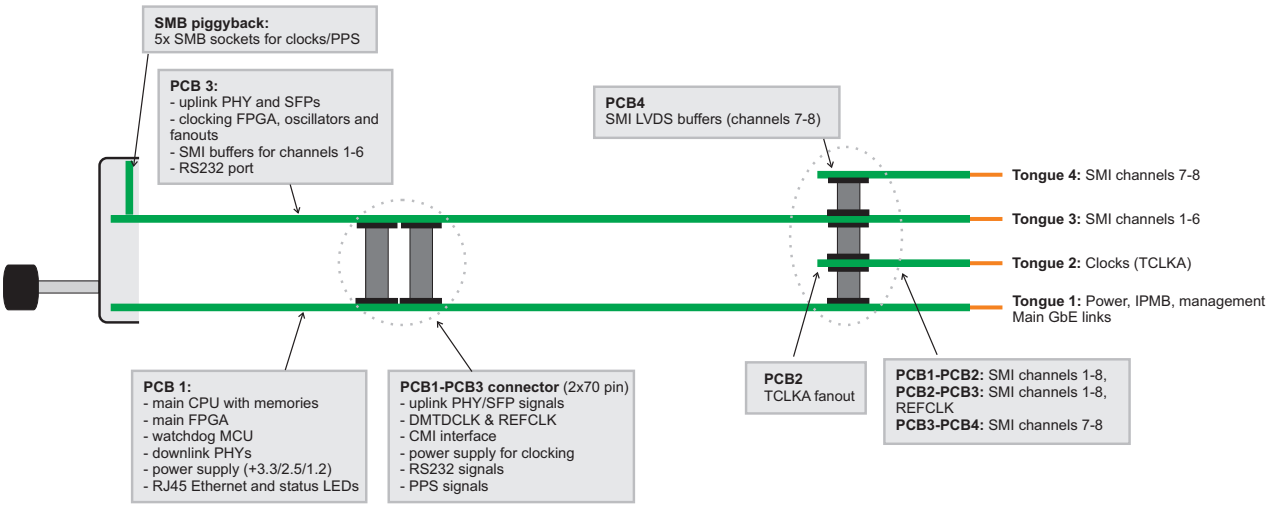



Figure 6: MCH PCB stackup

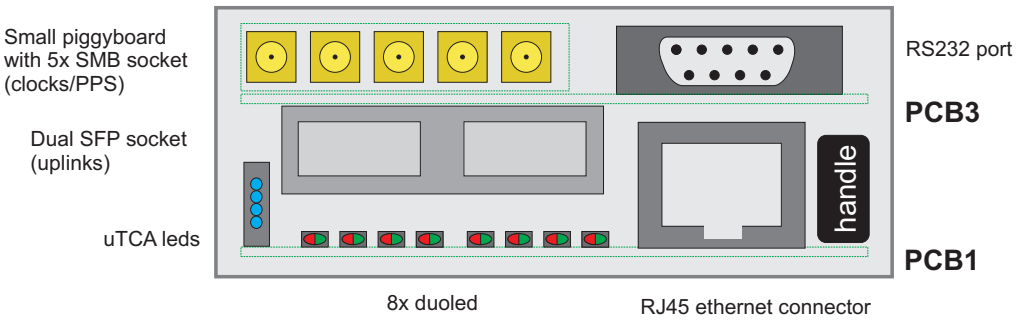



Figure 7: MCH front panel

## 5.2 CPU/FPGA boot process

To prevent bricking the switch by faulty remote firmware update, special boot procedure was developed. Blocks taking part in it are shown on figure 8.

The idea is to have two separate flash memories and two versions of firmware:

- Main firmware - currently working version, which can be updated at any time
- Failsafe firmware - minimal system and FPGA bitstreams required for device operation and main firmware update. This flash is not remotely writable.

Normal boot procedure is described below

1. Card gets management power (3.3V MP). Watchdog MCU powers up and enables main management power for MCU. Also it asserts FMAIN˙SEL signal and deasserts FBOOT˙SEL to force CPU to boot from main flash.

2. CPU firmware is loaded, special loader is launched which boots the FPGAs.

3. Device begins to operate.

In case we wanted to perform remote firmware update, we do:

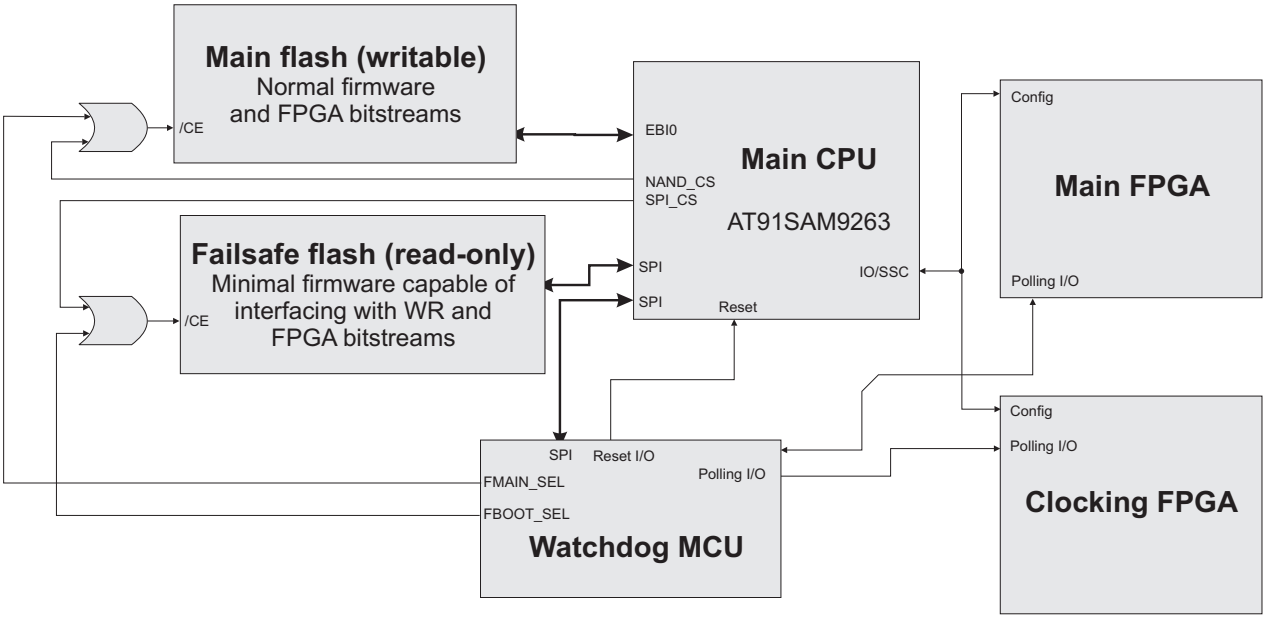



Figure 8: MCH boot process

1. Tell watchdog MCU that we are going to update the firmware and set up countdown timer. When timeout is exceeded, watchdog should reset main CPU and force it to boot using failsafe firmware.

2. Flash the new firmware and reboot

3. Login remotely again, do necessary chcecks (if new firmware works as we expected) and disable countdown timer in watchdog MCU.

## 5.3 Digital DMTD phase detector

For clock recovery/delay compensation circuit we've used Digital DMTD phase detector. Its main advantages are perfect linearity, simplicity (no need for analog parts except for single additional PLL/oscillator) and expandability. Thanks to the linerity, DDMTD can be used both as precise phase measurement device and as phase detector for PLL.

DDMTD shares basic concepts with traditional, analog DMTD (dual mixer time-domain) system. The big difference is that instead of analog mixers, we are using ordinary flip-flops sampling measured clocks. Mathematically, sampling operation can be expressed as multiplying input signal by series of Dirac pulses, but this approach can work on digital signals only.

Diagram of DDMTD is shown on figure 9.

The DDMTD consists of two flip-flops sampling input clocks (REFCLK, MEASCLK) using clock produced by helper PLL (DMTDCLK). The DMTDCLK frequency value is: $f_{DMTDCLK} = f_{REFCLK}\frac{N+1}{N}$ where N is DMTD divider - reasonably big integer number (like 10000).

If both input clocks are in phase and have identical frequencies, flip-flop outputs are also identical. If there is any difference between inputs, we can easily measure it by detecting

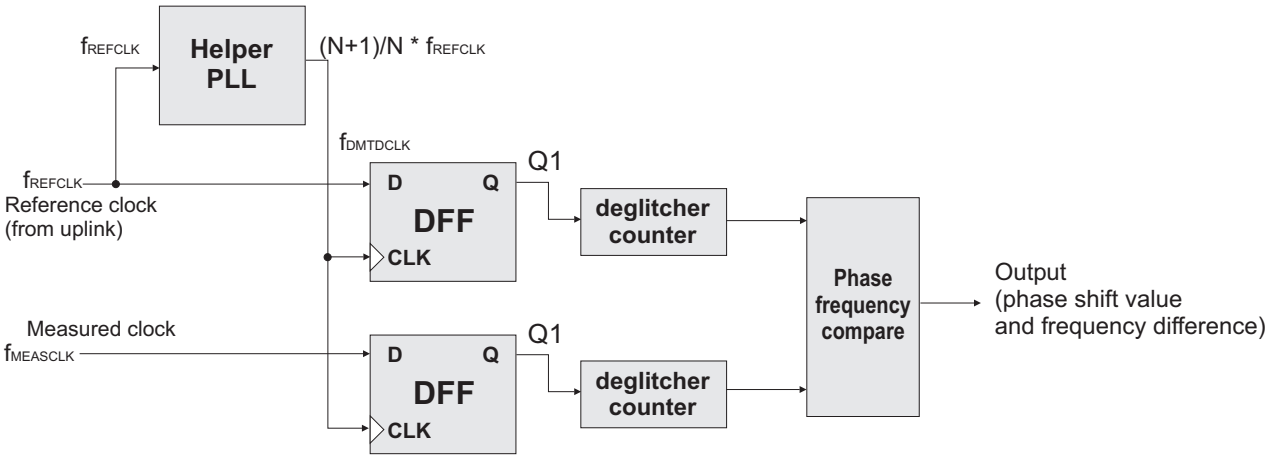



Figure 9: DDMTD detector

period and shift between output signals O1 and O2 using simple counter, as the O1, O2 frequency is divided by factor of N.

DDMTD simulation results (also with jittery clocks which manifest themselves as glitches on O1, O2 edges) are shown on figures 10, 11, 12.

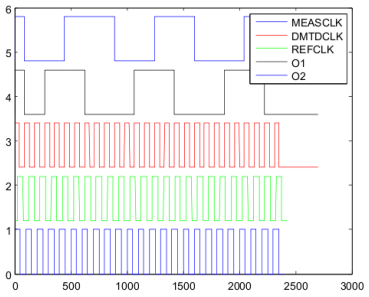



Figure 10: DDMTD simulation results - N=8, signals shifted by pi/2

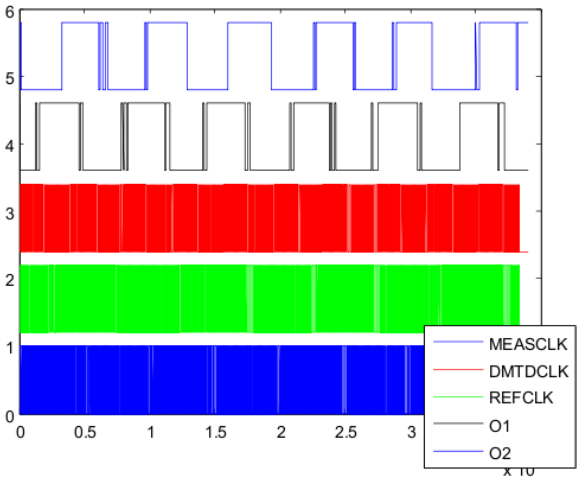



Figure 11: DDMTD simulation results - N=32, signals shifted by pi/2

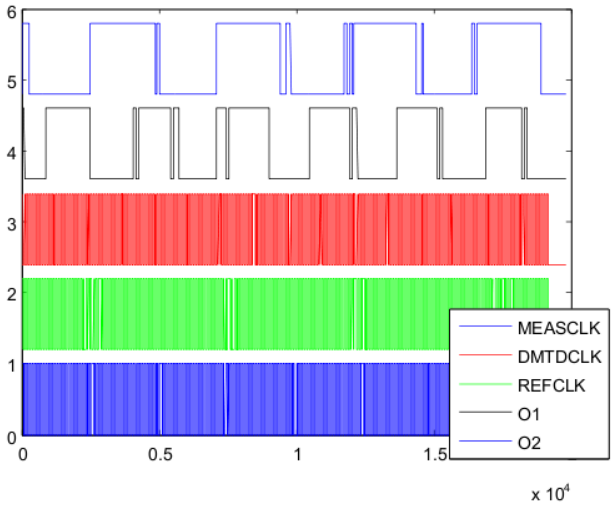



Figure 12: DDMTD simulation results - N=64, signals have different freqs

# 6 Interface descriptions

## 6.1 WR MCH CPU-FPGA bus

CPU-FPGA bus allows the CPU to:

- connect itself to WhiteRabbit network for handling PTP, IEEE802.1 and management protocols (SNMP, remote console, uTCA crate mgt)
- modify switch routing table. This feature is used by spanning-tree protocol and multicast group allocation. For ordinary, unicast traffic, routing table is updated automatically by MCH FPGA.
- read timestamp values, measure phase shifts and do the delay compensation
- manage switch ports
- use the switch management interface directly (e.g. for booting WR AMC cards)

Access to functions described above is available by memory-mapped registers. Preliminary register organization is shown on figure 13.

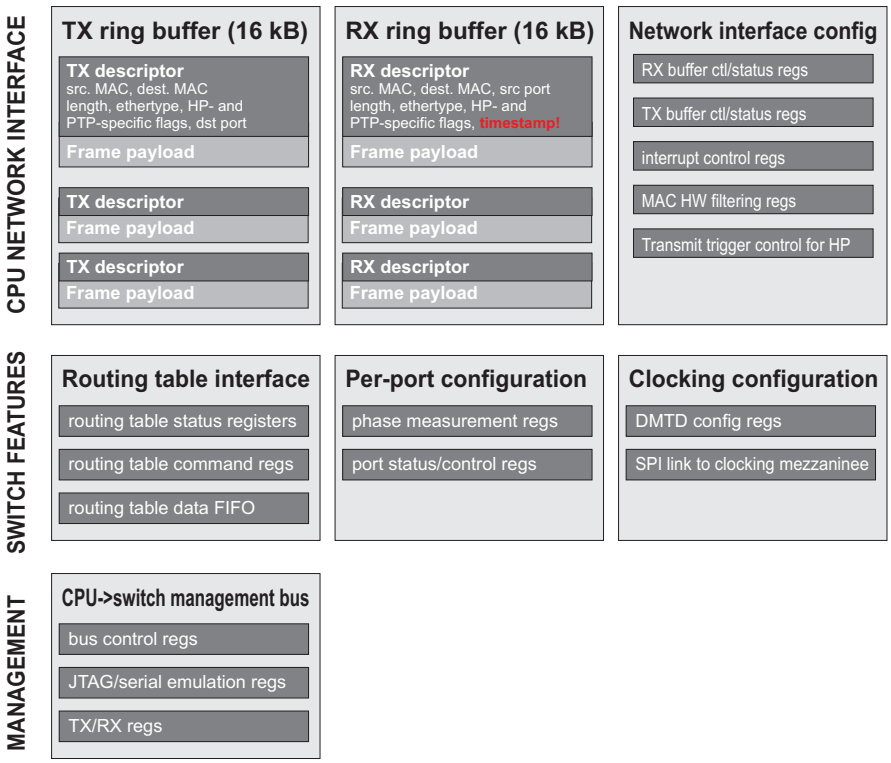



Figure 13: I/O register layout (preliminary)

### 6.1.1 Network interface controller (NIC)

The integrated NIC uses two 16-kilobyte ring buffers to transmit and receive data. In order to transmit a packet, following operations are required:

- check the amount of free space in TX ring buffer (and wait if there is not enough)
- fill the TX descriptor structure with source/destination MAC address, frame ethertype-/length and copy it to TX buffer
- copy frame payload into TX buffer
- set up interrupt control regs (if we want the end of transmission to be acknowledged)
- set up delayed send register (if we want NIC to start transmitting this frame at precise time)
- repeat these operations if there are more packets to send. After then, update TX descriptor length in TX control register.
- wait until packets are transmitted, either by polling TX status register or by waiting for interrupt.

Transmit operation can be completed in single CPU-driven DMA transfer (register setup and copying data).

Packet reception can be done using following algorithm:

- poll the RX status register for RX descriptor count or wait for RX completed interrupt
- disable interrupts
- copy the RX descriptor and frame payload from RX buffer
- re-enable interrupts
- extract data from RX descriptor
- free RX buffer space used by recently received descriptor by updating RX buffer control register.

Receive operation can also be done by single DMA transfer triggered by CPU after receiving interrupt request from NIC.

### 6.1.2 Routing table interface (RTI)

Through this interface, CPU can add, modify and remove entries from switch routing table:

- Adding entry is done by writing its data to RTI FIFO and triggering transfer by writing appropriate command to RTI command register. Then CPU shall poll the RTI status register or wait for interrupt, stating that requested change has been applied and routing tables in all WR AMCs in the crate are synchronized. Switch logic automaticly generates a hash value for each routing table entry. This value can be read by CPU from RTI status register and later used for modyfing or deleting the rule.
- Modifying entries is done in similar way as adding, except prior to sending updated entry data, CPU shall put the hash value of entry to be modified in RTI control register.
- Removing entries is done by putting hash value of rule to be deleted in RTI control register, executing deletion command by writing appropriate value to RTI control reg and polling status reg/waiting for interrupt.

### 6.1.3 Per-port configuration, phase measurement and clocking control

Using this interface, CPU can:

- Enable/disable each port features like: fragmentation, HP routing, synchronous operation, timestamping module operation, hub mode by writing to Port control register.
- Read phase measurement results from DMTD by accessing DMTD registers
- Program mainboard DMTD PLL (divider ratio, filter coeffs) by accessing DMTD registers
- Communicate with clocking mezzanine FPGA by accessing SPI FIFO registers.

### 6.1.4 Switch management interface (SMI)

CPU has limited SMI interface access for booting and diagnostics of WR AMC cards. Other SMI transfers (like timestamps, port IDs, routing table synchronization) are done by FPGA independently.

SMI can emulate both JTAG and passive serial bus. Also, it can provide raw, generic purpose access to FPGA pins connected to JTAG/serial ports and additional signals (if required).

## 6.2 Switch management interface bus

The switch management interface bus (SMI) is a slow (125 Mbit/sec) bus operating in parallel to main GbE backplane link. Its main purpose is providing communication between WR MCH FPGA and slave WR AMC FPGAs for data which cannot be multiplexed into Gigabit Ethernet link without risk of loosing determinism:

- packet send/receive timestamps
- packet source port IDs
- routing table updates
- sending PPS signal and UTC time to slave cards synchronously with TCLKA.

SMI bus operates synchronously to TCLKA reference clock. Data is sampled on rising clock edge. In order to achieve reliable symbol synchronization, SMI data is 8b10b-encoded. Example transfer is shown on figure 14.

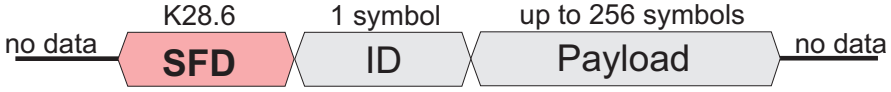



Figure 14: SMI frame structure

SMI frames have very simple structure, consisting of:

- 8b10b K28.6 sync character ('comma')

- 1-byte frame type/ID field
- up to 256 bytes of frame payload

For timestamps and port IDs, appropriate frame is sent using SMI simultaneously with packet data transmission via GbE link. Below is the list of basic SMI frame types:

- `SMI_PPS` - frame marking PPS signal pulse. The TCLK edge clocking in the first bit of this frame is the beginning of PPS period.
- `SMI_UTC` - current UTC time. Sent right after `SMI_PPS` to provide full time information.
- `SMI_SOURCEPORT` - frame containing source port information for currently transmitted packet. Used by WR MCH FPGA to update routing tables.
- `SMI_RXTIMESTAMP` - frame RX timestamp
- `SMI_TXTIMESTAMP` - frame transmit timestamp.
- `SMI_SWITCH` - switches SMI multiplexer between boot FPGA and main FPGA in WR AMC card.
- `SMI_RTADD`, `SMI_RTDEL`, `SMI_RTUPDATE` - modification of routing tables in WR AMC cards
- `SMI_WRITEREG`, `SMI_READREG` - direct access to FPGA registers (used for JTAG and passive serial booting)