WR Fabric Interface (WRF) informal specification version 0.2

Tomasz Włostowski/CERN BE-Co-HT

Table of Contents

1. Introduction

White Rabbit Fabric Interface (further referenced as WRF) is a system-on-chip interconnect for transporting Ethernet frames between various Ethernet- and WR-enabled FPGA peripherals. WRF provides a simple point-to-point 16-bit links with data format designed for easy parsing and modifying Ethernet frames without having to worry about IEEE802.xx compliance.

The major modules using WRF are:

- WhiteRabbit Endpoint (`wrsw_endpoint`)
- Switch NIC (`wrsw_nic`)
- Node mini-NIC (`wrsw_minic`)
- WhiteRabbit packet switching core (`wrsw_swcore`)

WRF provides following frame transport features:

- 16-bit data width (which corresponds to relatively slow 62.5 MHz clock rate at 1 Gbps)
- Frame field tagging using 4-bit control bus
- Half-word (16-bit)/byte-aligned transfers
- FIFO flow control (DREQ-DRDY style)
- Error indication and forced transfer abort
- Passing out-of-band information (timestamps, frame IDs, QoS data)

2. WRF link structure

Single WRF interface is an unidirectional, point-to-point link, so for a typical (duplex) Ethernet link one needs to create two WRF interfaces (one for outgoing and one for incoming packets). Each WRF link connects a single packet source to a single packet sink:
- WRF **packet source** is a port which outputs packets (for example, WR endpoint is a source of packets which are received from the fiber)
- WRF **packet sink** is a port which receives packets (for example, WR endpoint is a sink for packets to be transmitted to the fiber)

WRF ports on source and sink differ slightly, hence the source- or sink-specific signals will be described separately. A sample WRF link, connecting WR Endpoint to WR miNIC is shown on Fig. 1.

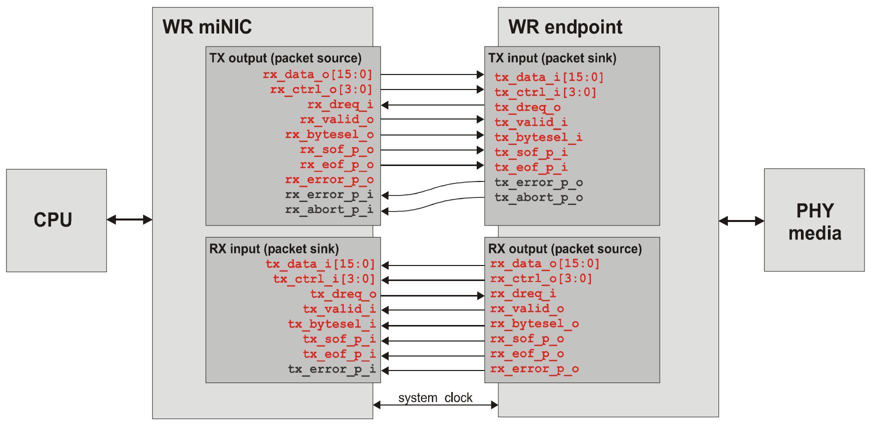

Fig.1. Sample WRF link connecting mini-NIC to WR endpoint.

3. WRF mandatory signals

Commonly used naming conventions (WR endpoint, miNIC and WRF SystemVerilog model):

– `rx_` prefix identifies ports of the packet source
– `tx_` prefix identifies ports of the packet sink
– `_p1` suffix indicates a single clock cycle (active HI) pulse

Table 1. Mandatory WRF signals

Name	Size	Direction	Source port	Sink port	Description
`data`	16	source ⟶ sink	`rx_data_o`	`tx_data_i`	Frame data bus
`ctrl`	4	source ⟶ sink	`rx_ctrl_o`	`tx_ctrl_i`	Control/tag bus
`bytesel`	1	source ⟶ sink	`rx_bytesel_o`	`tx_bytesel_i`	Byte/word select
`sof_p1`	1	source ⟶ sink	`rx_sof_p1_o`	`tx_sof_p1_i`	Start-of-frame indicator
`eof_p1`	1	source ⟶ sink	`rx_eof_p1_o`	`tx_eof_p1_i`	End-of-frame indicator
`dreq`	1	sink ⟶ source	`rx_dreq_i`	`tx_dreq_o`	Data Request line
`valid`	1	source ⟶ sink	`rx_valid_o`	`tx_valid_i`	Data Valid line
`rerror_p1`	1	source ⟶ sink	`rx_rerror_p1_o`	`tx_rerror_p1_i`	Packet source error indication

3.1. DATA

DATA is a 16-bit port, which outputs/inputs subsequent 16-bit frame words. The first byte is stored in the most significant part of the 16-bit data word. Value passed on DATA is correct and can be accepted by packet sink only when VALID line is active. When BYTESEL line is active, only the MSB part of DATA contains a valid frame byte. This can only be the last byte of the frame payload.

3.2. CTRL

CTRL bus is used to identify the type of data present on DATA lines.

Table 2. CTRL bus values (see `hdl/common/global_defs.vhd` for constant defs)

Name	Value	Description
`c_wrsw_ctrl_none`	0	No data tag, or the packet source doesn't identify current word of the frame as relevant for the MAC client (for example, the 0x8100 constant ethertype from VLAN-tagged frames) . When the source MAC address part of the packet header is tagged with `ctrl_none` value, WR endpoint automatically updates it with its own MAC address
`c_wrsw_ctrl_dst_mac`	1	Identifies destination MAC address part of frame header (first 3 words)
`c_wrsw_ctrl_src_mac`	2	Identifies source MAC address part of frame header (second 3 words)
`c_wrsw_ctrl_ethertype`	3	Identifies frame Ethertype (word 6 or 8 for 802.1q frames)
`c_wrsw_ctrl_vid_prio`	4	Identifies VLAN ID/PCP field from 802.1q header.

`c_wrsw_ctrl_tx_oob`	5	Identifies Out-Of-Band part of a frame to be transmitted by WR endpoint, containing the frame ID. The TX OOB structure is described in p. ….
`c_wrsw_ctrl_rx_oob`	6	Identifies Out-Of-Band part of a frame received by WR endpoint, containing the source physical port ID and RX timestamp. RX OOB structure is described in p. ….
`c_wrsw_ctrl_payload`	7	Identifies the frame payload (e.g. everything which doesn't belong to frame header). Frame payload does not include the FCS field, which is never transmitted on WRF

3.3. BYTESEL

When BYTESEL line is active, DATA bus bits [15:8] contain the last single byte of transferred frame. Such situation can occur only when the length of frame is odd.

Warning: WRF does not allow for interleaving byte- and word- sized transfers. BYTESEL can be used only for the last byte of the frame payload.

3.4. DREQ / VALID

DREQ and VALID lines provide a simple flow control mechanism. Packes sink activates the DREQ line in order to:

- request the next word of the frame currently being transferred
- allow the packet source to initiate a transfer of a new frame (e.g. assert SOF_P1)

When packet source detects that DREQ line is active, it shall provide the subsequent frame data word on DATA/CTRL bus and assert the VALID line. If the packet source can't provide any data at the moment, the VALID line shall stay low.

DREQ line is also used to request another frame if no frame is currently being transferred. Packet source can only assert SOF_P1 line when DREQ is active.

Warning: VALID, DATA, CTRL and BYTESEL, SOF_P lines must always be registered.

3.5. SOF_P1 and EOF_P1 lines

SOF_P1 line provides a single-clock pulse before the beginning of a new frame transfer. Packet source can assert SOF_P1 only when the packet sink requests data by activating DREQ line.

First frame word transfer must never occur when SOF_P1 is active. In such situation, the packet sink shall ignore the state of VALID line.

EOF_P1 line indicates the end of currently transferred frame with **no errors**. When EOF_P1 is active, there may still be a valid data word present on DATA and CTRL, so the packet sink must monitor VALID line even when EOF_P1 is active.

3.6. *RERROR_P1*

RERROR_P1 indicates an errorneous packet termination when the error occurred in the packet source. For example, WR Endpoint receiver can assert RERROR_P1 when:

- frame CRC is invalid
- a PCS encoding error occured (invalid 8b10b code, misalignment, sync loss, etc.)
- frame has been corrupted due to buffer overrun

RERROR_P1 behaves in identical way as EOF_P1. These signals are mutually exclusive (e.g. only one of them can be active in the same clock cycle).

4. WRF optional signals

Table 3. Optional WRF signals

Name	**Size**	**Direction**	**Source port**	**Sink port**	**Description**
`terror_p1`	1	sink ⟶ source	`rx_terror_p1_i`	`tx_terror_p1_o`	Packet sink error indication
`idle`	1	source ⟶ sink	`rx_idle_o`	`rx_idle_i`	Source idle flag
`tabort_p1`	1	source ⟶ sink	`rx_tabort_p1_o`	`tx_tabort_p1_i`	Source-initiated transfer abort
`rabort_p1`	1	sink ⟶ source	`rx_rabort_p1_i`	`tx_rabort_p1_o`	Sink-initiated transfer abort

4.1. *TERROR_P1*

Single-cycle pulse on TERROR_P1 during packet transfer indicates that an error occurred during processing the frame by the packet sink. For example, WR Endpoint transmitter will assert TERROR_P1 when:

- TX buffer underrun has occured
- Frame was too long

Upon reception of TERROR_P1 event, packet source shall decide whether to drop or retransmit the frame.

4.2. *IDLE*

When active, the packet source is doing nothing. Can be used for debugging purposes.

4.3. *TABORT_P1*

Single-cycle pulse tells the packet sink to abort the reception of current frame. In case of WR endpoint, it means that the frame must be deliberately corrupted (so the remote side won't receive it properly).

Note: TABORT_P1 is mutually exclusive with EOF_P1 and ERROR_P1

4.4. *RABORT_P1*

Single-cycle pulse tells packet source that the packet sink is unable to finish reception of currently transferred packet. After RABORT_P1 event, packet source shall immediately terminate the transfer by asserting RABORT_P1 and eventually retransmit the interrupted packet later (the exact behavior is to be defined by the core developer – in case of WR endpoint, all the packets are dropped as long as RABORT_P1 is active).

5. OOB formats

5.1. *TX OOB format*

TX OOB is used to tag the transmitted frame with an unique 16-bit ID, used by WR endpoints to queue TX timestamps for multiple physical ports.

Table 4. TX OOB format

OOB word	**Description**
1st	16-bit frame ID tag. Must be unique in a reasonably long period of time (in our case it's the TX timestamp polling interval)

5.2. *RX OOB format*

RX OOB encodes the orginating physical port identifier to which the packet came and the value of packet's RX timestamp.

Table 5 . RX OOB format

OOB word	**Description**
1st	Bits [15:11]: physical port ID (value of ECR:PORTID field in endpoint configuration regs)
2nd	Bits [15:12]: 4 least significant bits of falling edge timestamp counter Bits [11:0] 12 most significant bits of rising edge (main) TS counter
3rd	16 least significant bits of rising edge (main) TS counter

6. Example frame transfer

Figure 2 shows a typical successful frame transfer with associated RX OOB fields. The frame is 33 bytes-long (so technically it's not a valid frame because it's too short), and it consists of:

- destination MAC: 00:01:02:03:04:05
- source MAC: 06:07:08:09:0a:0b
- ethertype: 0x0c0d
- payload: increasing byte values from 0x0e to 0x20. **Note that frames are transferred without FCS checksum, which is always calculated and stripped by the WR endpoint**.
- RX OOB block, with port ID = 0x1, rising edge timestamp = 0x1234567 and falling edge timestamp = 0x6

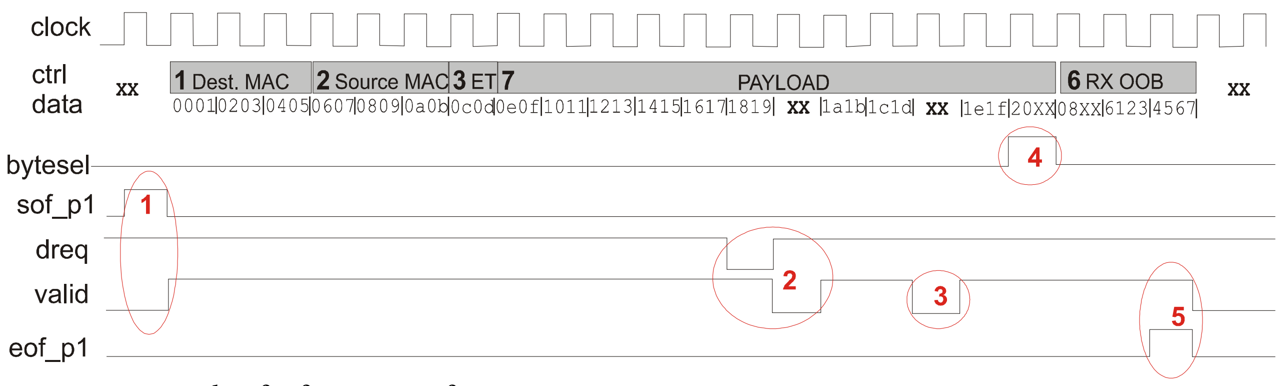
clock
ctrl
data
xx
1 Dest. MAC
2 Source MAC
3 ET
7
PAYLOAD
6 RX OOB
xx
0001|0203|0405|0607|0809|0a0b|0c0d|0e0f|1011|1213|1415|1617|1819| xx |1a1b|1c1d| xx |1e1f|20XX|08XX|6123|4567|
bytesel
sof_p1
dreq
valid
eof_p1
1
2
3
4
5

Figure 2. Example of a frame transfer over WRF

Some extra explanation for the drawing:

1. **Start-of-frame event** – SOF_P1 pulse must always occur at least one clock cycle before the transfer of the first data word (so VALID = 0 when SOF_P1 == 1)
2. **Flow control forced by packet sink**: the packet sink was unable to accept a data word, so it had set the DREQ line to 0. The packet source didn't send a data word in the following clock cycle (VALID was 0).
3. **Flow control forced by packet souce**: the source didn't have any data in its output buffer, so VALID line was set to 0 for a single cycle.
4. **Odd-sized frame**: the last word of the payload contains the last, 33rd byte of the frame which is indicated by activating BYTESEL line.
5. **End-of-frame event** – EOF_P1 is active, indicating the end of current frame transfer. Note that there's the last, 3rd word of RX OOB present on DATA/CTRL lines, so VALID line is still active.

7. WRF SystemVerilog model

A WRF bus-functional model available in the SVN repo (`trunk/hdl/sim/fabric_emu_new`) with some examples:

- `fabric_emu.sv` provides a module with 1 packet source/sink and a simple API for sending/receiving frames (see tasks `send()` and `receive()`)
- `fabric_emu_tap.sv` provides a fabric emulator using Linux TAP device to feed the WRF with real packets coming from TAP network interface.
- `fabric_emu_demo.sv` and `fabric_demo_tap.sv` show how to use these emulators. For the TAP interface setup instructions, see the comments at the beginning of the file.