SINTEF

Address:

NO-
NORWAY

Enterprise /VAT No:

Project memo

AXI_S chain link switch. IP module for FPGAs. Documentation.

Subtitle

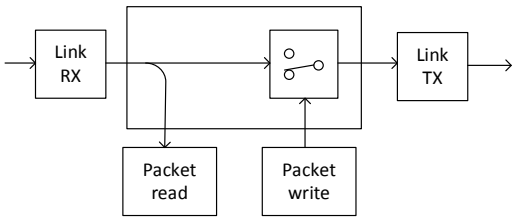
VERSION	DATE
1.0	2016-06-24

AUTHOR(S)
Kjell Ljøkelsøy

CLIENT(S)	CLIENTS REF.

PROJECT NO.	NO. OF PAGES AND APPENDICES:
	8

ABSTRACT
This memo contains documentation of a FPGA IP module, including functional description, lists of parameters, signals and registers. This IP module is as connecting element in an AXI_S signal chain. It is intended for use in communication link organised as a daisy chain of point to point gigabit fiber links.



- Incoming data frames are forwarded to the outgoing link, and out to a local packet reader.
- Data frames from a local packet writer are sent to the outgoing link via a packet switch.
- The frame changeover switch is governed by the end of packet flags. When the ongoing transfer is finished, it switched to the firs node that has data waiting to be transferred.
- Start of incoming packages are flagged, so they can be timestamped.

PREPARED BY	SIGNATURE
Kjell Ljøkelsøy	

APPROVED BY	SIGNATURE
Giuseppe Guidi	

PROJECT MEMO NO.	CLASSIFICATION
AN xx.12.16	Unrestricted

Document history

VERSION	DATE	VERSION DESCRIPTION
1.0	2016-06-06	Initial version

SINTEF

Table of contents

1 Introduction

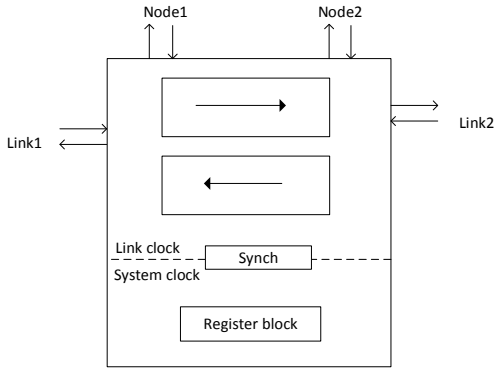


Figure 1 Chain link switch overview

This IP module connects two point to point links, in order to form a bus system with daisy chain structure. The module is unidirectional, so two modules are required for a bidirectional link.

A point to point link can consist of a gigabit serial link, either using copper cable or optical fiber, using Aurora8b10b protocol.

Interface to the switch module is by AXI stream interface, so any link type can be used.
Connections to local packing and unpacking blocks is also by AXI stream interface.
A processor bus interface is used for configuration and status handling.

2 Description

This module contains configuration and management circuits, an AXI register array, and two unidirectional changeover switches, one for each direction.

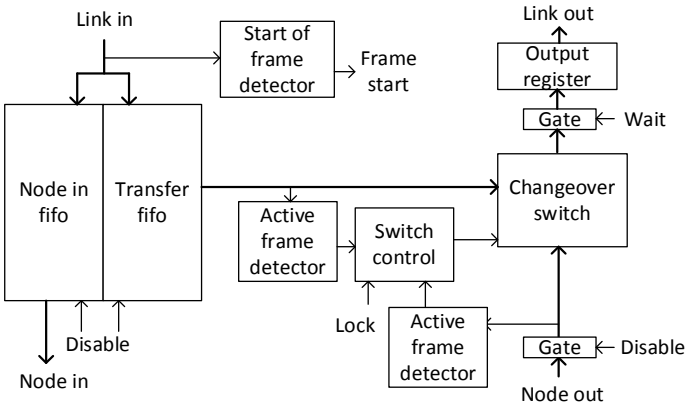


Figure 2 Unidirectional link switch

An unidirectional changeover switch contains:

- Two fifos with a common input signal, one for node input, one for transferring data from one link to the changeover switch for the next link.
- An output register with a flow control switch at its input.

- A line changeover switch. It operates only when there are idle interval between frames. It is locked at the resent input when multiple frames are transferred without any idle intervals between them.

- Line switch control. Changing is only performed when the presently engaged link is inactive.

- Start of frame detector at the line input. It generates frame start pulse that can be used for synchronisation of link nodes.

- Frame active detectors at line and node inputs to the changeover switch. The interval between the axis_tlast and the next axis_tready pulse is declared as idle.

The input FIFOs are ready to receive data words anytime, so interfacing to an Aurora gigabit transceiver core, which does not read the tready –signal does not give any problems, unless the buffer overflows.

The Start of frame pulse is transferred across the clock boundary, so it is synchronous too the AXI bus clock.

The register array contains status and configuration registers, accessible on the AXI bus. The register array runs on the bus clock, so the status and control signals are transferred across the clock boundary via using synchronisation registers that suppresses metastability.

Fifo fill level for two different thresholds are flagged. This can be used for flow control with hysteresis.

- Node in FIFO disable flag flushes and disables the node input FIFO so it discards any incoming data. This ensures that an unused node input does not interfere with any input flow control function.

- Transfer FIFO disable flag flushes and disables the transfer FIFO so it discards any incoming data. This blocks any transfer between the two channels, so they can be used independently of each other.

- Node out disable flag disables node input to the changeover switch, and flushes the node out input by setting the tready signal active.

The link flush signals sets the node and transfer input FIFO disable signals and the output register for each link. This means that a link flush signal disables the transfer FIFOs in both directions, while node in and out transmission on the adjacent link is not affected. The control signals can be set individually from the control register.

The Fifos can be configured to pause write to fifo when one of the fifos is full, using the tvalid-tready hansdshake mechanism. This does not work when connected to an Aurora IP core, as its RX AXI stream output does not use the tready signal. The switch will then stall with a partially transferred frame locking the changeover switch and blocked fifo inputs.

The disable signals for the fifos and the node out input acts immediately, so they may amputate any ongoing messages, both when turned on and off. A message in the middle of transfer when the disable signal is set will be amputated because it loses its tlast signal, which indicates the end of the frame. The next incoming message will then be spliced onto the amputated frame so the next one is being lost too.

The changeover switch can be set to be locked in off- state. It will stay in its present state until any ongoing frame transfer is finished. This gives soft turnoff without lost/amputated frames.

A link_out_wait signal pauses data transfers out from the output register. It can be used for flow control. It does not cause any data loss, unless the fifos overflows.

3 Parameters

```
LINK_FIFO_SIZE                  : integer := 2048;  -- Size of transfer fifo and node input fifo
LINK_FIFO_FILLED_FLAG_LEVEL_A : integer := 256;   -- threshold for fifo fill level signal A
LINK_FIFO_FILLED_FLAG_LEVEL_B : integer := 128;      -- threshold for fifo fill level signal B
CONFIG_REGISTER_DEFAULTVALUE   : integer:= 0; -- Sets default value in register.
-- Parameters of Axis interface
C_AXIS_DATA_WIDTH                    : integer := 32;

-- Parameters of Axi Slave Bus Interface S_AXI
C_S_AXI_DATA_WIDTH                   : integer   := 32;
C_S_AXI_ADDR_WIDTH                   : integer   := 4
```

4 Ports

```
-- Signals synchronous to system clock.
  frame1_start_pulse              :  out std_logic := '0'; -- High for one clock cycle at the start of a new frame at the link input.
  frame2_start_pulse              :  out std_logic := '0';

  -- Signals synchronous to link clock.
  fifo_filled_flag_a1   : out std_logic;              -- fifo fill level indicator. '1': Above threshold A
  fifo_filled_flag_a2   : out std_logic;
  fifo_filled_flag_b1    : out std_logic;             -- fifo fill level indicator. '1': Above threshold B
  fifo_filled_flag_b2   : out std_logic;

  stream_switch_state_1 : out std_logic_vector (2-1 downto 0);   -- State of the changeover switch.  -- 0: idle 1: node out 2: transfer fifo
  stream_switch_state_2 : out std_logic_vector (2-1 downto 0);

  link1_wait                      : in std_logic := '0'; --  flow control '1' halts transmission on link out
  link2_wait                      : in std_logic := '0';

  link1_flush                     : in std_logic := '0'; --   '1' Flushes trasfer fifo, input fifo and output.
  link2_flush                     : in std_logic := '0'; --   '1' Flushes trasfer fifo, input fifo and output.

  -- clocks, reset
  axis_aclk                       : in std_logic;                 -- link clock
  axis_aresetn                    : in std_logic;

  s_axi_aclk                      : in std_logic;                 -- system clock.
  s_axi_aresetn                   : in std_logic;

 -- AXI stream interfaces. Synchronous to axis_aclk. Contains signals:   tdata, tkeep, tlast, tvalid, tready
--  (Note: AXIS tready  signal is not used on Aurora RX stream outputs)
link_in1_axis
link_in2_axis
link_out1_axis
link_out2_axis
node_in1
node_in2
node_out1
node_out2

-- AXI lite interface . Synchronous to s_axi_aclk
```

5 Timing constraints

Signals crossing clock domain boundaries requires timinc constraints to be specified in order to avoid timing errors:

```
set_false_path -from * -to [get_pins {*/*/*regarray/U0/axi_rdata_reg*/D}]
```

The number of "*/" in the constraint depends on the actual design hierarchy. This can be determined to build without any constraint, and use the timing error report to determine the actual constraints.

6 Registers

```
//-****************************************************************************
//     AXI_S chain link switch
//     Author: Kjell Ljøkelsøy Sintef Energy.      2016.
//-****************************************************************************

#define AXI_S_CHAIN_LINK_SWITCH_STATUSREG(BASEADDR)    *(volatile unsigned int*)(BASEADDR + 4* (0))

     #define    LINK_OUT_WAITING_1_BITNR                0 // 1: Transfer of data at link out is blocked.  Flow control. No data loss.

     #define    ACTIVE_TRANSFER_FRAME_1_BITNR           2  // 1: Frame from the link input is being transferred to link out or is waiting.
     #define    ACTIVE_NODE_OUT_FRAME_1_BITNR           3  // 1: Frame from node is being transferred or is waiting.

     #define    SWITCH_STATE_1_START_BITNR              4   // State of the changeover switch.  2 bit signal: 0: idle 1: node out 2: transfer fifo

     #define    TRANSFER_FIFO_FILLED_A1_BITNR           8  // 1:Transfer fifo from link to changeover switch is filled to flag threshold level A.
     #define    TRANSFER_FIFO_FILLED_B1_BITNR           9  //  1: Transfer fifo from link to changeover switch is filled to flag threshold level B.
     #define    NODE_FIFO_FILLED_A1_BITNR               10 // 1: Node input fifo is filled to flag threshold level A.
     #define    NODE_FIFO_FILLED_B1_BITNR               11 // 1: Node input fifo is filled to flag threshold level A.

     #define    TRANSFER_FIFO_FULL_1_BITNR              12 // 1: Transfer fifo from link to changeover switch is full.
     #define    NODE_FIFO_FULL_1_BITNR                  13 // 1: Node input fifo is full.

     #define    LINK_FLUSH_1_BITNR                      15  // 1: Hardware input signal is set. Flushes both node and transfer fifo.

     #define    LINK_OUT_WAITING_2_BITNR                16

     #define    ACTIVE_TRANSFER_FRAME_2_BITNR           18
     #define    ACTIVE_NODE_OUT_FRAME_2_BITNR           19

     #define    SWITCH_STATE_2_START_BITNR              20

     #define    TRANSFER_FIFO_FILLED_A2_BITNR           24
     #define    TRANSFER_FIFO_FILLED_B2_BITNR           25

     #define    NODE_FIFO_FILLED__A2BITNR               26
     #define    NODE_FIFO_FILLED_B2_BITNR               27

     #define    TRANSFER_FIFO_FULL_2_BITNR              28
     #define    NODE_FIFO_FULL_2_BITNR                  29

     #define    LINK_FLUSH_2_BITNR                      31 // hardware input

#define AXI_S_CHAIN_LINK_SWITCH_CONFIGREG(BASEADDR)    *(volatile unsigned int*)(BASEADDR + 4* (1))

     #define LINK_OUT_WAIT1_BITNR                      0     // 1: Blocks transfer of data at link out.          Flow control. No data loss.
     #define LINK_OUT_WAIT2_BITNR                      16
```

```
#define SET_CHANGEOVER_SWITCH1_TO_NONE_BITNR  1 // 1:   When ongoing frame transfer is finished the changeover switch is locked.
#define SET_CHANGEOVER_SWITCH2_TO_NONE_BITNR  17 // None of the inputs are active. Soft disable, No data loss.

#define NODE_OUT1_DISABLE_BITNR : integer     8  // 1: Node out input is disabled, and flushed. Aborts ongoing transfers immediately.
#define NODE_OUT2_DISABLE_BITNR : integer     24   // No tlast flag is sent. Next message is appended to amputated message, so both are lost.

#define NODE_IN1_FIFO_DISABLE_BITNR           9  // 1: node input FIFO is flushed.  Hard disable, Ongoing transfer is abored.
#define NODE_IN2_FIFO_DISABLE_BITNR           25 // No tlast flag is sent. Next message is appended to amputated message, so both are lost.

#define LINK1_2_IN_OUT_TRANSFER_DISABLE_BITNR 10   // 1: link transfer FIFO is flushed.  Ongoing transfer is aborted. No tlast flag is sent.
#define LINK2_1_IN_OUT_TRANSFER_DISABLE_BITNR 26   //  Next message is appended to amputated message, so both are lost.

#define FIFO1_INPUT_HANDSHAKE_BITNR           12   //'0' No handshake. overwrites, flushes fifo. '1' pauses writing (tready handshake)
#define FIFO2_INPUT_HANDSHAKE_BITNR           28   // (The Aurora IP does not use tready handshake at its RX AXI stream interface.)


#define NODE_FIFO_FILL_LEVEL1_REGNR(BASEADDR)       *(volatile unsigned int*)(BASEADDR + 4* (2))  // Fill level of input fifo
#define NODE_FIFO_FILL_LEVEL1_REGNR(BASEADDR)       *(volatile unsigned int*)(BASEADDR + 4* (3)

#define TRANSFER_FIFO_FILL_LEVEL_1_2_REGNR(BASEADDR)  *(volatile unsigned int*)(BASEADDR + 4* (4))  // Fill level of transfer fifo
#define TRANSFER_FIFO_FILL_LEVEL_2_1_REGNR(BASEADDR)  *(volatile unsigned int*)(BASEADDR + 4* (5))
```

Technology for a better society

www.sintef.no