

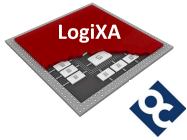
# ESoCL

## Ethernet Switch on Configurable Logic

### Design Document

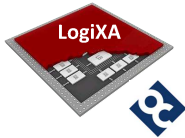
Copyright (C) LogiXA and OPENCORES.ORG

This source file may be used and distributed without restriction provided that this copyright statement is not removed from the file and that any derivative work contains the original copyright notice and the associated disclaimer. This source file belongs to free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version. This source is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details. You should have received a copy of the GNU Lesser General Public License along with this source; if not, download it from <http://www.opencores.org/lgpl.shtml>



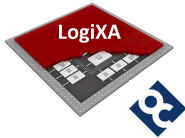
### HISTORY

Version	Description	Author	Date
D00	First draft release	L.Maarsen	05.05.2014
D01	Roadmap added, see chapter 6	L.Maarsen	07.05.2014
D02	Detailed description added, see chapter 3	L.Maarsen	05.07.2014

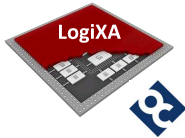


## Table of contents

HISTORY .....	2
1. INTRODUCTION .....	6
2. GET STARTED .....	8
2.1 IP Configuration.....	8
2.2 IP Memory Initialisation .....	9
2.2.1 VLAN Table .....	9
2.2.2 Boot ROM .....	9
2.3 Simulation Macro .....	9
3. DESIGN DESCRIPTION .....	10
3.1 Block diagram .....	10
3.2 Interconnect busses .....	11
3.3 Clocks.....	11
3.4 Host interface .....	12
3.5 Ethernet port.....	12
3.5.1 Port interface .....	12
3.5.2 Port storage.....	13
3.5.3 Port processor .....	14
3.6 Data bus arbiter .....	15
3.7 Search Bus arbiter .....	15
3.8 Search engine .....	15
3.8.1 DA Processing.....	16
3.8.2 SA Processing.....	16
3.8.3 Aging Control.....	17
4. REGISTER DESCRIPTION.....	18
4.1 Register Map .....	18
4.2 Registers for Ethernet Port .....	19



4.2.1	PORT_PROC_VLAN_TABLE_CTRL .....	19
4.2.2	PORT_PROC_OUTBOUND_DONE_COUNT .....	19
4.2.3	PORT_PROC_OUTBOUND_DROP_COUNT .....	19
4.2.4	PORT_PROC_INBOUND_DONE_COUNT .....	20
4.2.5	PORT_PROC_INBOUND_DROP_COUNT .....	20
4.2.6	PORT_PROC_SEARCH_DONE_COUNT .....	20
4.2.7	PORT_PROC_SEARCH_DROP_COUNT .....	20
4.2.8	PORT_PROC_STAT_CTRL .....	21
4.2.9	PORT_MAL_VLAN_CTRL .....	21
4.2.10	PORT_MAL_STAT_CTRL .....	21
4.2.11	PORT_MAC_STAT_CTRL .....	22
4.3	Registers for Control & Status .....	23
4.3.1	CTRL_SCRATCH .....	23
4.3.2	CTRL_STAT_CTRL .....	23
4.3.3	CTRL_VERSION .....	23
4.3.4	CTRL_ID .....	24
4.4	Registers for Data Bus Arbiter .....	24
4.4.1	DB_ARB_PORT_DISABLE .....	24
4.4.2	DB_ARB_PORT_WEIGHT .....	24
4.5	Registers for Search Bus Arbiter .....	25
4.5.1	SB_ARB_PORT_DISABLE .....	25
4.5.2	SB_ARB_PORT_WEIGHT .....	25
4.6	Registers for Search Engine .....	26
4.6.1	SEARCH_ENGINE_SA_OVERLOAD_COUNT .....	26
4.6.2	SEARCH_ENGINE_SA_DROP_COUNT .....	26
4.6.3	SEARCH_ENGINE_STAT_CTRL .....	26
4.7	Registers for Test Bench .....	27
4.7.1	RGMII_PORT_ENABLE .....	27
5.	SIMULATION ENVIRONMENT .....	28
5.1	Block diagram .....	28
5.2	Folder structure .....	29
5.3	Macro's .....	29
5.4	Execute a simulation .....	30
5.5	Scripts .....	30
5.5.1	Scripts for Initialization .....	31



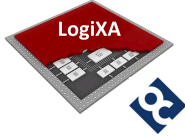
5.5.2	Scripts for basic packet flow .....	31
5.5.3	Scripts for prioritized packet flow.....	35
5.5.4	Scripts for Output congestion control .....	37
5.5.5	Scripts for Input congestion control .....	39
5.5.6	Scripts for search engine SA.....	40
5.5.7	Scripts for Search engine DA.....	43
5.5.8	Scripts for VLAN Tag Editing.....	44
5.5.9	Scripts for boot ROM.....	45
5.5.10	Scripts for PHY management .....	46
6.	ROADMAP .....	47

### List of figures

Figure 1:	block diagram of ESoCL .....	10
Figure 2:	block diagram of test bench .....	28

### List of tables

Table 1:	Parameters in ESoCL configuration file.....	8
Table 2:	performance indication ESoCL .....	12
Table 3:	Register map of ESoCL.....	18
Table 4:	Modelsim macro's .....	30



### 1. INTRODUCTION

This document describes the Ethernet Switch on Configurable Logic, further referred to as ESoCL.

The ESoCL is a configurable Ethernet Switch that can be used to design Ethernet Switch functionality into FPGA based solutions to achieve a more flexible and integrated solution. The ESoCL supports up to 16 MAC ports, the MAC ports support an (R)MII or an (R)GMII MAC interface towards an external Ethernet PHY. Each MAC port has its own serial PHY management interface.

The ESoCL is a VLAN capable Ethernet Switch. Untagged packets received by a port are tagged by the port default VLAN ID before they are processed. After switching an untagged packet to the destination port(s) the default tag is removed. Tagged packets can be re-tagged by the port default VLAN ID before or after they are processed further.

The ESoCL is a self-learning Ethernet Switch, the source address of incoming packets are stored or renewed in the internal MAC Address table, together with the associated VLAN ID. The MAC address table is monitored by an aging mechanism that removes expired MAC addresses.

The ESoCL can operate in unmanaged and managed mode. For unmanaged applications the ESoCL has an integrated boot ROM that can be used to configure the Ethernet Switch after reset is de-asserted. For managed applications the ESoCL has a generic, asynchronous, memory mapped interface that can be used by a processor platform to configure the Ethernet Switch. Managed applications with challenging start-up requirements related to the Ethernet Switch functionality can use the boot ROM as well.

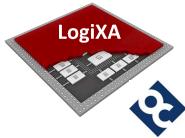
#### Main Features

- Ethernet Switch with up to 16 ports
- Shared total switching capacity of 6-9 Gbps<sup>1</sup>
- Serial PHY management on each port
- Integrated Ethernet MACs with (R)MII or (R)GMII interface from Altera, Xilinx<sup>2</sup> or Open Cores<sup>2</sup>.
- MAC address table with up to 8192 entries
- Configurable address learning and aging mechanism
- VLAN tagged frames, according to IEEE 802.1Q, up to 4096 VLAN ID's

---

<sup>1</sup> Depends on target device and average packet size

<sup>2</sup> Altera is supported, a Xilinx and a full Open Cores version are on the roadmap



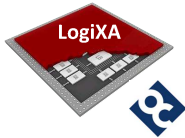
## Ethernet Switch on Configurable Logic

---

- Generic host or Open Cores Wish Bone<sup>3</sup> interface
- Unmanaged and managed operation
- Initialization by boot ROM
- Extensive set of counters

---

<sup>3</sup> Open Cores Wish Bone interface is on the roadmap



## 2. GET STARTED

Download the ESoCL design from one of these locations:

- SVN repository at <http://opencores.org/ocsvn/esoc/trunk>
- Direct download of latest version at <http://opencores.org/download/esoc>

Store the working copy into a local folder, further referred to as `<drive>:\<folder>`

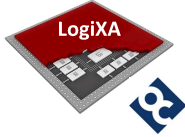
### 2.1 IP Configuration

Use the file `<drive>:\<folder>\Sources\logixa\package_esoc_configuration.vhd` to configure the following parameters of the ESoCL:

Parameter	Type	Default value	Valid Range	Description
esoc_version	integer	latest version	0 ... $2^{16}-1$	Version number of ESoCL build. Version number will change after a major update.
esoc_release	integer	latest release	0 ... $2^{16}-1$	Release number of ESoCL build. Release number will change after a major update.
esoc_mode	esoc_modes	simulation	normal, simulation	Mode of ESoCL, use simulation to <i>speed</i> up simulations, use <i>normal</i> to build a version for a target.
esoc_brom_mode	esoc_brom_modes	disabled	enabled, disabled	Boot ROM mode, use <i>enabled</i> when internal Boot ROM is required, use <i>disabled</i> when no internal Boot ROM is required, see chapter 2.2 for Boot ROM memory initialization.
esoc_port_count	integer	8	2 ... 16	Number of Ethernet ports on the ESoCL.
esoc_search_engine_col_depth	integer	7	0 ... 7	Number of entries to be checked after a hashed DA or hashed SA pointer doesn't give a positive result in the MAC Address table.

Table 1: Parameters in ESoCL configuration file





## 2.2 IP Memory Initialisation

### 2.2.1 VLAN Table

The content of the VLAN Table is defined in the file:

`<drive>:\<folder>\Sources\altera\esoc_ram_nkx1\esoc_ram_4kx1.mif`

The structure of the VLAN Table memory is

- Address N: Member of VLAN N when set to 1, else not member of VLAN
- Address N+1: Member of VLAN N+1 when set to 1, else not member of VLAN

Each port has a VLAN Table with the initial content determined by this file. The content of the VLAN table memory of each port can be individually modified by the host processor through the register `PORT_PROC_VLAN_TABLE_CTRL` or by the Boot ROM.

### 2.2.2 Boot ROM

The content of the Boot ROM is defined in the file:

`<drive>:\<folder>\Sources\altera\esoc_rom_nkx32\esoc_rom_2kx32.mif`

The structure of the Boot ROM memory is

- Address N: ESoCL Register Address
- Address N+1: ESoCL Register Data

If the Register Address is `0xFFFFFFFF` the initialization of the ESoCL by Boot ROM is terminated and the operational mode is started if initialization by Boot ROM was successful. The register `CTRL_STAT` can be used to see if Boot ROM is used and if initialization by Boot Rom was successful.

## 2.3 Simulation Macro

The simulation environment is driven by macro's. The macro

`<drive>:\<folder>\Simulation\Modelsim\init.do` requires one path setting if the folder structure of the ESoCL working copy is not changed after download.

The macro requires one argument when executed in Modelsim, the working environment. A working environment is e.g. Home or Work, the parameter `path_project_files` must be set to `<drive>:\<folder>` for all applicable working environments.

## 3. DESIGN DESCRIPTION

### 3.1 Block diagram

The block diagram of the ESoCL is shown below.

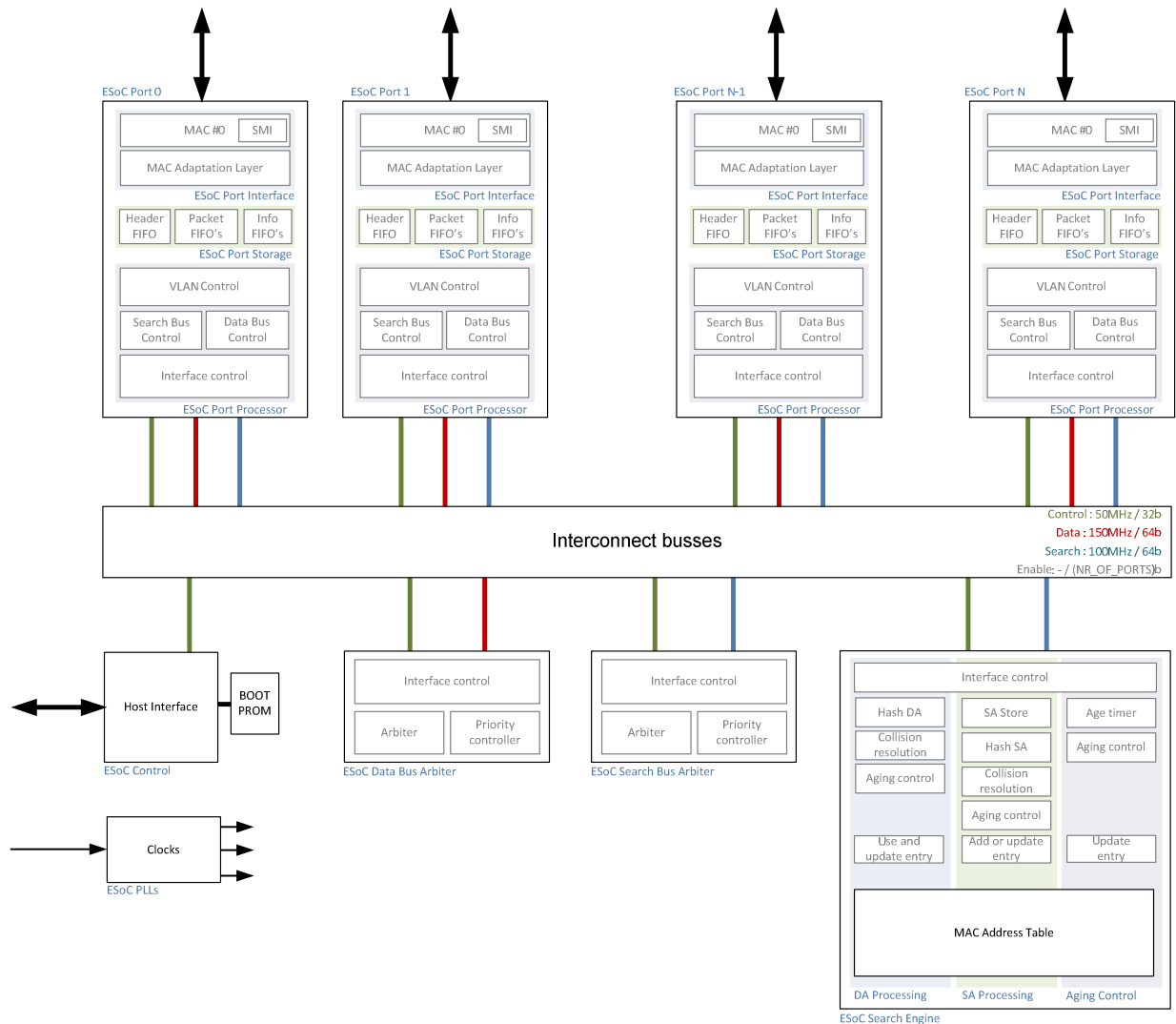
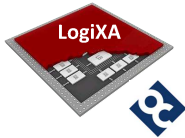


Figure 1: block diagram of ESoCL

The following units are described in the subsequent chapters:

- Interconnect buses
- Clocks
- Host interface
- Ethernet port
  - Port interface



- Port storage
- Port processor
- Data bus arbiter
- Search Bus arbiter
- Search engine
  - DA Processing
  - SA Processing
  - Aging control

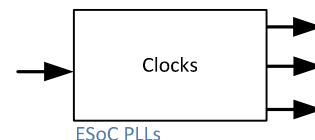
### 3.2 Interconnect busses

All the units inside the ESoCL can communicate with each other through the following three interconnect busses:

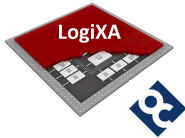
- Control bus, a 32 bit point-to-multi point bus to access all status and control registers inside the units from the host.
- Data bus, a 64 bit point-to-multi point bus to transfer packets from the source port to the destination port(s).
- Search bus, a 64 bit point-to-point bus to transfer search requests and replies between the source port and the search engine.
- Enable bus, a point-to-multi point bus to enable destination port(s) from the source port.

### 3.3 Clocks

The ESoCL uses one input clock of 50MHz and generates different internal clocks by means of PLL's. The three interconnect busses operate at different frequencies and depending on the FPGA target these bus speeds can be scaled up or need to be scaled down. The ESoCL is currently operating at the following speeds:



Interconnect bus	Speed	ESoCL Performance
Control	50MHz	-
Data	150MHz	5.8Gbps - 64B packet size 7.6Gbps - 500 B packet size 7.8Gbps - 1000 B packet size 7.9Gbps - 1500 B packet size  <i>Data bus performance is calculated, performance indication assumes 100% throughput at all ports, all packets the same size.</i>
Search	100MHz	14286ksps - direct hit in MAC table 10000ksps - hit in MAC table after three linear search actions 7143ksps - hit in MAC table after three linear search actions

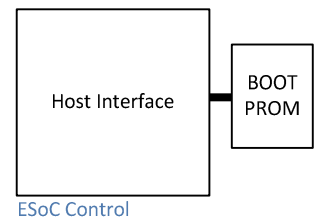


		<p>A port 100% throughput correspond with</p> <ul style="list-style-type: none"> <li>1953kpps / 64B packet size</li> <li>250kpps / 500B packet size</li> <li>125kpps / 1000B packet size</li> <li>83kpps / 1500B packet size</li> </ul> <p><i>Search bus performance is calculated, the performance indication assumes 100% load at all ports, all packets the same size. kpps = Kilo Packets Per Second; ksps = Kilo Search requests Per Second</i></p>
--	--	--

Table 2: performance indication ESoCL

### 3.4 Host interface

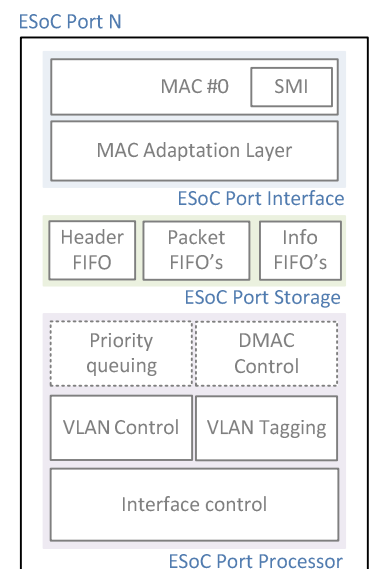
The host interface unit provides the interface between the ESoCL and an external host. The host interface is an asynchronous variable latency interface. The external host must wait after initiating a read or write cycle upon the de-assertion of the signal ESOC\_WAIT before a new read or write cycle is started.



After the ESoCL is started the Host interface unit shall start with accessing the internal Boot ROM and initializing the ESoCL with the content from this Boot ROM. After the initialization from Boot ROM is complete, indicated to the external host by the signal ESOC\_BOOT\_COMPLETE, the external host can access all the registers inside the ESoCL. The result of the Boot ROM initialization can be read from register CTRL\_STAT\_CTRL, see 4.3.2.

### 3.5 Ethernet port

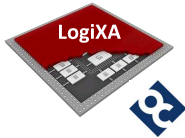
The Ethernet port unit provides the interface and processing between the Ethernet RGMII port towards the external Ethernet PHY and the internal interconnect busses. This Ethernet port unit has three sub-units and is instantiated multiple times according to the esoc\_port\_count parameter, see chapter 2.1.



#### 3.5.1 Port interface

The Port Interface sub-unit includes the Triple Speed Ethernet MAC unit of Altera and a MAL unit – MAC Adaptation Layer unit – to have a common interface between any MAC unit and the internal architecture of the ESoCL.

The MAC unit is configured with internal FIFO's, a Serial Management Interface for PHY management and a RGMII PHY Interface towards the external PHY. However, other configurations for the MAC layer are possible. In chapter 6 can be found which future extensions are planned for integration of other MAC units, like the MAC layer from Xilinx or



Opencores. After power-up the MAC need to be configured by the internal Boot ROM or an external host to a.o. enable the transmit and receive path, set maximum packet length, enable CRC32 removal and insertion by MAC, enable padding bytes removal and enable the discard erroneous packets feature.

### ***Inbound, packet stream from network to ESoCL***

The interface between the MAC unit and the MAL unit is a 32 bit streaming interface. As soon as there is a new packet completely received and accepted by the MAC it will be streamed to the MAL unit. The MAL unit shall write into three inbound FIFO's. These three inbound FIFO's are part of the Port Storage unit and forms the interface between the MAL unit and the Port Processor unit.

The MAL unit shall write the complete packet header into the inbound *Header FIFO* as soon as the header is received, it shall write the complete packet into the inbound *Packet FIFO* and when the complete packet is written in this inbound *Packet FIFO* it shall write packet information like packet length, VLAN ID (if present) into the inbound *Info FIFO*. The header shall be written into the *Header FIFO* before the complete packet is received, to start already the Search Bus request and the search action to find out the destination port.

The MAL unit shall replace the VLAN ID by the port default VLAN ID if the packet is tagged with VLAN ID 0, a QoS packet, or when the force port default VLAN ID on inbound feature is enabled through the register `PORT_MAL_VLAN_CTRL`, see chapter 4.2.9.

### ***Outbound, packet stream from ESoCL to network***

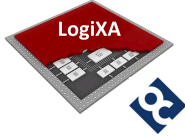
As soon as the Port Storage unit has written a packet information entry in the outbound *Info FIFO* the MAL unit shall read the info entry and use it to read the complete packet from the outbound *Packet FIFO* and stream it to the MAC unit.

The MAL unit shall replace the VLAN ID by the port default VLAN ID if the packet is tagged and the force port default VLAN ID on outbound feature is enabled through the register `PORT_MAL_VLAN_CTRL`, see chapter 4.2.9.

## **3.5.2 Port storage**

The Port Storage unit provides FIFO's for packet storage, FIFO for packet header storage, FIFO's for packet info storage, bus width translation and bus speed translation between the Port Interface unit and Port Processor unit.

The streaming interfaces between the MAC unit and the MAL unit have a 32 bit bus width and operate at the control clock, see 3.3, while the Port Processor unit operates with a data bus width of 64 bit and operate at the data clock, see 3.3. The bus width expansion and the bus speed increase are necessary to provide serious switching performance.



### 3.5.3 Port processor

#### ***Inbound, packet stream from network to ESoCL***

The Port Processor unit shall check the VLAN ID of the packet and the VLAN memberships of the inbound port when header information is written in the Header FIFO of the Port Storage unit by the MAL Unit. The Port Processor unit shall request access to the Search Bus when the inbound port is a member of the packet VLAN, else the packet will be dropped. The Port Processor unit shall request for the Search Bus immediately when the packet is not tagged. VLAN Memberships can be set through the register PORT\_PROC\_VLAN\_TABLE\_CTRL, see chapter 4.2.1.

The Port Processor unit shall transfer the source MAC address, the Destination MAC address and the VLAN ID of the incoming packet to the Search Engine unit when the Search Bus is assigned to the Port Processor unit by the Search Bus Arbiter unit. The result of the search action shall be stored in the Search Result FIFO. The requesting port itself is masked from the search result.

The Port Processor unit shall request access to the Data Bus when packet information is written in the Info FIFO of the Port Storage unit by the MAL Unit. The Search result for the corresponding packet is then already available in the Search Result FIFO. The Port Processor unit shall transfer the packet to the destination port(s) when the Data Bus is assigned to the requesting port by the Data Bus Arbiter unit.

The Port Processor unit shall prepend an additional word is to inform the destination port(s) about the length of the packet, the source port of the packet and the VLAN ID if the packet is tagged.

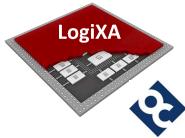
The Port Processor unit shall enable the destination ports – based upon the search result - through the Port Enable bus. The Search Engine unit shall already skip a destination port from the Search result if a destination port indicates an overload situation to the Search Engine unit.

#### ***Outbound, packet stream from ESoCL to network***

The Port Processor unit is continuously listening to the Port Enable bus and when enabled by a source port it shall use the prepended packet data with information about the packet for further processing.

The Port Processor unit shall check the VLAN ID of the packet and the VLAN memberships of the outbound port and accept the real packet when the outbound port is a member of the packet VLAN, else the packet will be dropped. VLAN Memberships can be set through the register PORT\_PROC\_VLAN\_TABLE\_CTRL, see chapter 4.2.1.

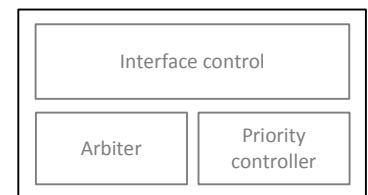
The Port Processor unit shall write the packet in the packet FIFO of the Port Storage Unit and when the packet is completely written in the packet FIFO it shall write an entry into the



info FIFO of the port Storage unit. The MAL unit shall process the packet as soon as the info entry is written. If the packet FIFO has insufficient space to store the whole packet the Port Processor unit shall stop writing the packet FIFO and shall write an entry in the info FIFO with the DROP bit enabled and the actual number of bytes written. The MAL unit shall drop this incomplete packet.

### 3.6 Data bus arbiter

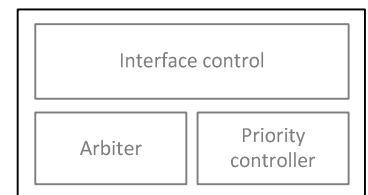
The Data Bus Arbiter unit is based upon the round robin principle and after power-up each port of the ESoCL has the same priority. The priority of each port can be changed through register DB\_ARB\_PORT\_WEIGHT, see chapter 4.4.2. If an Ethernet port requires the bus it shall send a Data Bus request to the Data Bus Arbiter unit and wait for the Data Bus grant.



ESoC Data Bus Arbiter

### 3.7 Search Bus arbiter

The Search Bus Arbiter unit is based upon the round robin principle and after power-up each port of the ESoCL has the same priority. The priority of each port can be changed through register SB\_ARB\_PORT\_WEIGHT, see chapter 4.5.2. If an Ethernet port requires the bus it shall send a Search Bus request to the Data Bus Arbiter unit and wait for the Search Bus grant.

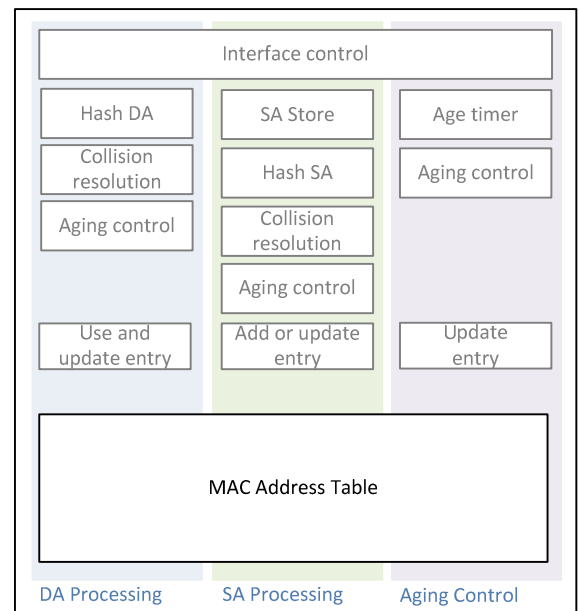


ESoC Search Bus Arbiter

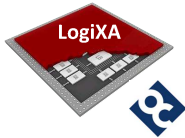
### 3.8 Search engine

The Search Engine unit performs three tasks to maintain the MAC Address Table. The three tasks are:

- DA Processing, search for Destination MAC addresses and return the destination ports to the requesting port, if there is a hit.
- SA Processing, learn Source MAC addresses seen on inbound ports if not already present in the MAC address table.
- Age control, remove MAC addresses after their age is expired.
- 



ESoC Search Engine



### 3.8.1 DA Processing

The Search Engine unit shall hash the Network Interface Controller Identifier part of the destination MAC address – the lowest 24 bit of the 48 bit MAC address - that is provided by the requesting port over the Search Bus into a 10 bit value that is used as a pointer to an entry inside the MAC address table.

Due to the hashing mechanism there is a chance that multiple MAC addresses result in the same pointer value, called a collision. Therefore the search engine shall use the pointer to the first entry as a starting point and perform a linear search if there is no hit on the first entry. The length of the linear search action is determined by the parameter `ESOC_SEARCH_ENGINE_COL_DEPTH`, see chapter 2.1. The total search area inside the MAC address table is called the collision buffer.

The entry data is read and compared with the Destination MAC address and VLAN ID and if there is a hit and the entry is valid, which means not out of date, the outbound port is returned to the requesting port.

If there is no hit within the collision buffer the Search Engine unit shall return all outbound ports (broadcast). The requesting port itself shall avoid that the packet is send back on its own port.

### 3.8.2 SA Processing

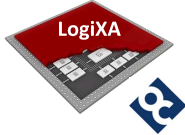
Each port shall also transfer the source MAC address to the Search Engine unit in addition to destination MAC address and VLAN ID when it has a search request. The Search Engine unit shall store these source MAC addresses into a FIFO because the Source Address learn processes is interleaved with the Aging Control process. Source MAC addresses are dropped if this FIFO is full. However, Source Address learning has a higher priority than Aging Control. The number of dropped source MAC addresses can be read from register `SEARCH_ENGINE_SA_OVERLOAD_COUNT`, see chapter 4.6.1.

If there are no Aging Control actions active on an entry the Search Engine unit shall process the source MAC address that are collected in the FIFO and when the FIFO is empty the Aging Control shall continue. The Search Engine unit shall hash the Network Interface Controller Identifier part of the source MAC address into a 10 bit value that is used as a pointer to an entry inside the MAC address table.

Due to the hashing mechanism there is a chance that multiple MAC addresses result in the same pointer value. Therefore the search engine shall use the pointer to the first entry as a starting point and perform a linear search if there is no hit on the first entry. The length of the linear search action is determined by the parameter `ESOC_SEARCH_ENGINE_COL_DEPTH`, see chapter 2.1.

The entry data is read and compared with the Source MAC address and VLAN ID and if there





is a hit and the entry is valid the entry will be updated. Updating an entry means writing the inbound port number, because it can be different and setting the Aging Control flag, because this Aging Control flag could be reset by the Aging mechanism, see chapter 3.8.3.

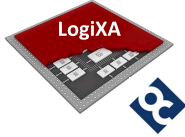
If there is no hit within the collision buffer the Search Engine unit shall write the first empty entry inside the collision buffer with the new source MAC address, VLAN, inbound port number and set the valid and Aging Control flag. The Search Engine unit already learned the position of the empty entry during the linear search action. The source MAC address is dropped if there is no empty entry in the collision buffer. The number of dropped source MAC addresses can be read from register `SEARCH_ENGINE_SA_DROP_COUNT`, see chapter 4.6.2.

### 3.8.3 Aging Control

The Aging Control unit is enabled by default, but register `SEARCH_ENGINE_STAT_CTRL`, see chapter 4.6.3, can be used to disable the Aging Control unit or to modify the maximum age of the MAC addresses. Aging Control is executed when the aging timer expires. From that moment the Aging Control unit shall read each entry from the MAC Address table and check the Aging Control flag of each valid entry. The Aging timer is restarted and the Aging Control unit shall wait for the next trigger when the end of the MAC Address table is reached.

The Aging Control flag is set when the entry is written for the first time or updated by the Source MAC address learn process, see chapter 3.8.2. If the Aging Control flag is set the Aging Control unit shall reset the flag, but keep the entry valid. If the flag is already reset – so in one aging period the particular MAC address is not seen - the Aging Control unit shall reset the valid flag, which means that the MAC address is no longer valid.

The actions on the MAC Address table by the Aging Control unit are interleaved with the actions by the Source Address Processing unit, the latter has a higher priority. The Aging Control unit is paused if source addresses arrived and continued if all source addresses are processed.



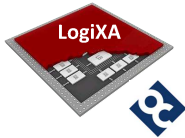
### 4. REGISTER DESCRIPTION

The ESoCL register set is description the following chapters. All registers can be initialised by the Boot ROM, if the Boot ROM is part of the build.

#### 4.1 Register Map

Register	Address	Reset value	Chapter
PORT_PROC_VLAN_TABLE_CTRL	Port_NR * 0x800 + 0x197	0x00000000	4.2.1
PORT_PROC_OUTBOUND_DONE_COUNT	Port_NR * 0x800 + 0x196	0x00000000	4.2.2
PORT_PROC_OUTBOUND_DROP_COUNT	Port_NR * 0x800 + 0x195	0x00000000	4.2.3
PORT_PROC_INBOUND_DONE_COUNT	Port_NR * 0x800 + 0x194	0x00000000	0
PORT_PROC_INBOUND_DROP_COUNT	Port_NR * 0x800 + 0x193	0x00000000	4.2.5
PORT_PROC_SEARCH_DONE_COUNT	Port_NR * 0x800 + 0x192	0x00000000	4.2.6
PORT_PROC_SEARCH_DROP_COUNT	Port_NR * 0x800 + 0x191	0x00000000	4.2.7
PORT_PROC_STAT_CTRL	Port_NR * 0x800 + 0x190	0x00000000	0
PORT_MAL_STAT_CTRL	Port_NR * 0x800 + 0x180	0x00000000	4.2.9
PORT_MAC_STAT_CTRL	Port_NR * 0x800 + 0x00...0xFF	0x00000000	4.2.11
CTRL_SCRATCH	0x8003	0x00000000	4.3.1
CTRL_STAT_CTRL	0x8002	0xC0000008	4.3.2
CTRL_VERSION	0x8001	0x00010000	0
CTRL_ID	0x8000	0x71022001	4.3.4
DB_ARB_PORT_DISABLE	0x8807	0x00000000	4.4.1
DB_ARB_PORT_WEIGHT	0x8800	0x00000000	4.4.2
SB_ARB_PORT_DISABLE	0x880F	0x00000000	4.5.1
SB_ARB_PORT_WEIGHT	0x8808	0x00000000	4.5.2
SEARCH_ENGINE_SA_OVERLOAD_COUNT	0x8812	0x00000000	4.6.1
SEARCH_ENGINE_SA_DROP_COUNT	0x8811	0x00000000	0
SEARCH_ENGINE_STAT_CTRL	0x8810	0x8000012C	4.6.3
RGMII_PORT_ENABLE	0xFF00	0x00000000	4.7.1

Table 3: Register map of ESoCL



## 4.2 Registers for Ethernet Port

### 4.2.1 PORT\_PROC\_VLAN\_TABLE\_CTRL

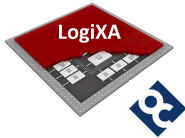
Register    PORT_PROC_VLAN_TABLE_CTRL    Base Address: port * 0x0800 Offset: 0x197 Reset value: 0x00000000			
Bit	Name	Description	Access
31	VLAN_CTRL_TYPE	Type of operation inside the VLAN table of this port 0 = read entry from VLAN table 1 = write entry in table  The host can get the membership of a particular VLAN by writing the VLAN ID and set the VLAN_CTRL_TYPE to read The following read operation of this register will provide the VLAN membership.	RW
30	VLAN_MEMBER	VLAN Membership 0 = port is not member of VLAN identified by the field VLAN_ID 1 = port is member of VLAN identified by the field VLAN_ID	RW
29..12		Not used	
11..0	VLAN_ID	VLAN ID to control in table. By default all ports are member of VLAN ID 0.	W

### 4.2.2 PORT\_PROC\_OUTBOUND\_DONE\_COUNT

Register    PORT_PROC_OUTBOUND_DONE_COUNT    Base Address: port * 0x0800 Offset: 0x196 Reset value: 0x00000000			
Bit	Name	Description	Access
31..0	OUTBOUND_DONE_COUNT	Counter represents the number of successful packet transfers to this port.	R

### 4.2.3 PORT\_PROC\_OUTBOUND\_DROP\_COUNT

Register    PORT_PROC_OUTBOUND_DROP_COUNT    Base Address: port * 0x0800 Offset: 0x195 Reset value: 0x00000000			
Bit	Name	Description	Access
31..0	OUTBOUND_DROP_COUNT	Counter represents the number of dropped packets at this port, reason to drop: ☐ This port is not a member of the packet's VLAN. ☐ Outbound FIFO is full	R



#### 4.2.4 PORT\_PROC\_INBOUND\_DONE\_COUNT

Register PORT_PROC_INBOUND_DONE_COUNT			
		Base Address: port * 0x0800 Offset: 0x194 Reset value: 0x00000000	
Bit	Name	Description	Access
31..0	INBOUND_DONE_COUNT	Counter represents the number of successful packet transfers from this port to destination port(s).	R

#### 4.2.5 PORT\_PROC\_INBOUND\_DROP\_COUNT

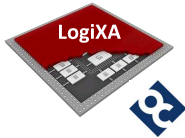
Register PORT_PROC_INBOUND_DROP_COUNT			
		Base Address: port * 0x0800 Offset: 0x193 Reset value: 0x00000000	
Bit	Name	Description	Access
31..0	INBOUND_DROP_COUNT	Counter represents the number of dropped packets at this port, reason to drop: <input checked="" type="checkbox"/> Search engine did not return a destination port(s). The reason for this is that the destination port(s) are congested. <input checked="" type="checkbox"/> Inbound FIFO is full	R

#### 4.2.6 PORT\_PROC\_SEARCH\_DONE\_COUNT

Register PORT_PROC_SEARCH_DONE_COUNT			
		Base Address: port * 0x0800 Offset: 0x192 Reset value: 0x00000000	
Bit	Name	Description	Access
31..0	SEARCH_DONE_COUNT	Counter represents the number of successful search actions. A search action is executed when the port is member of the packets VLAN, if tagged, else a search action is always executed.	R

#### 4.2.7 PORT\_PROC\_SEARCH\_DROP\_COUNT

Register PORT_PROC_SEARCH_DROP_COUNT			
		Base Address: port * 0x0800 Offset: 0x191 Reset value: 0x00000000	
Bit	Name	Description	Access
31..0	SEARCH_DROP_COUNT	Counter represents the number of dropped search actions, reason to drop: <input checked="" type="checkbox"/> Receiving port is not member of the packets VLAN, if tagged, else a search action is always executed. Search action is not executed.	R



#### 4.2.8 PORT\_PROC\_STAT\_CTRL

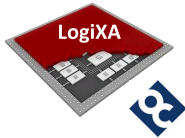
Register		PORT_PROC_STAT_CTRL		Base Address: port * 0x0800 Offset: 0x190 Reset value: 0x00000000	
Bit	Name	Description	Access		
31..0		Not used			
1..0	STATE	Port state 11 = Forwarding (normal operation) 10 = Learning (not forwarding) 01 = Blocked/Listening 00 = disabled	RW		

#### 4.2.9 PORT\_MAL\_VLAN\_CTRL

Register		PORT_MAL_VLAN_CTRL		Base Address: port * 0x0800 Offset: 0x181 Reset value: 0x00000001	
Bit	Name	Description	Access		
31	FORCE_OUTBOUND	Force default VLAN ID for tagged packets on outbound traffic 0 = do not force default VLAN ID for tagged packets 1 = force default VLAN ID for tagged packets	RW		
30	FORCE_INBOUND	Force default VLAN ID for tagged packets on inbound traffic 0 = do not force default VLAN ID for tagged packets 1 = force default VLAN ID for tagged packets	RW		
29..16		Not used			
15..13	PCP	Priority Code Point	RW		
12	CFI	Canonical Format Indicator	RW		
11..0	VID	Default VLAN ID of port. Used when incoming packet (UC/MC/BC) is not tagged. Packet itself will not be tagged, this default VLAN ID is used during the search operation and when a packet is transferred from source to destination port(s).	RW		

#### 4.2.10 PORT\_MAL\_STAT\_CTRL

Register		PORT_MAL_STAT_CTRL		Base Address: port * 0x0800 Offset: 0x180 Reset value: 0x00000001	
Bit	Name	Description	Access		
31..4		Not used			

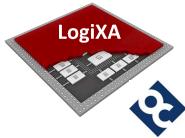


## Ethernet Switch on Configurable Logic

3	XONGEN	<p>Generation of a pause frame with a 0 pause quanta.            0 = do not generate pause frame            1 = generate pause frame</p> <p>☒ Ignored when the xon_gen bit in the MAC command_config register is set to 1.            ☒ Absent when the MAC Enable full duplex flow control option is turned off.</p>	RW
2	XOFFGEN	<p>Generation of a pause frame with a pause quanta configured in the MAC pause_quant register.            0 = do not generate pause frame            1 = generate pause frame</p> <p>☒ Ignored when the xon_gen bit in the MAC command_config register is set to 1.            ☒ Absent when the MAC Enable full duplex flow control option is turned off.</p>	RW
1	MAGICWAKE	<p>If the MAC function is in the power-down state, the MAC function asserts this signal to indicate that a magic packet has been detected and the node is requested to restore its normal frame reception mode.            0 = no magic packet detected            1 = magic packet detected</p> <p>☒ This signal is present only if the MAC Enable magic packet detection option is turned on.</p>	R
0	MAGICSLEEP	<p>Assert this active-low signal to put the node into a power-down state. If magic packets are supported - the MAC MAGIC_ENA bit in the command_config register is set to 1 - the receiver logic stops writing data to the receive FIFO buffer and the magic packet detection logic is enabled.</p> <p>0 = enable power down state            1 = disable power down state</p> <p>☒ This signal is present only if the MAC Enable magic packet detection option is turned on.</p>	RW

### 4.2.11 PORT\_MAC\_STAT\_CTRL

Register		PORT_MAC_STAT_CTRL		Base Address: port * 0x0800 Offset: 0x00 - 0xFF Reset value:	
Bit	Name	Description	Access		
		All MAC registers are documented in the user guide of the third party MAC			



### 4.3 Registers for Control & Status

#### 4.3.1 CTRL\_SCRATCH

Register	CTRL_SCRATCH		Base Address: 0x8000 Offset: 0x3 Reset value: 0x00000000
Bit	Name	Description	Access
31..0	SCRATCH	Scratch registers for test purposes	RW

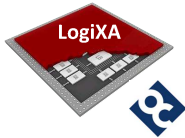
#### 4.3.2 CTRL\_STAT\_CTRL

Register	CTRL_STAT_CTRL		Base Address: 0x8000 Offset: 0x2 Reset value: 0xC0000008*
Bit	Name	Description	Access
31	PLL_LOCK2	PLL Lock status 0 = not locked 1 = locked	R
30	PLL_LOCK1	PLL Lock status 0 = not locked 1 = locked	R
29	BOOTINIT	Boot Initialization status, can be read after the boot complete output is asserted 0 = boot initialization successful 1 = boot initialization failed, corrupt rom data, device status unknown!	R
28	BOOTROM	Boot from rom 0 = disabled 1 = enabled  The boot from rom option is enabled/disabled before building the design, this bit can be used to check the status boot rom feature.	R
27..8		Not used	
7..0	PORTS	Number of ports, depends on settings before build	R

\*Reset value depends on ESOC\_BROM\_MODE parameter in ESocL configuration file, see chapter 2.1, and Boot initialization status

#### 4.3.3 CTRL\_VERSION

Register	CTRL_VERSION		Base Address: 0x8000 Offset: 0x1 Reset value: 0x00010000
Bit	Name	Description	Access
31..16	VERSION	ESOC Version changes after adding functionality	R
15..0	RELEASE	ESOC Release changes after bug fixes	R



### 4.3.4 CTRL\_ID

Register		CTRL_ID	Base Address: 0x8000 Offset: 0x0 Reset value: 0x71022001	
Bit	Name	Description	Access	
31..12	MANID	Manufacturer ID	R	
11..0	DEVID	Device ID	R	

## 4.4 Registers for Data Bus Arbiter

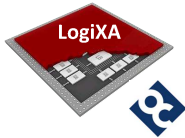
### 4.4.1 DB\_ARB\_PORT\_DISABLE

Register		DB_ARB_PORT_DISABLE	Base Address: 0x8800 Offset: 0x7 Reset value: 0x00000000	
Bit	Name	Description	Access	
n..0	DISABLE	Port n enable/disable internal data bus access 0 = enable 1 = disable  0 <= n <= 15, port enable/disable control for internal data bus access. For test purposes only, use the Ethernet MAC TX and RX enable/disable control bits for enabling and disabling a port during operational use.	RW	

### 4.4.2 DB\_ARB\_PORT\_WEIGHT

Register		DB_ARB_PORT_WEIGHT	Base Address: 0x8800 Offset: 0x0 Reset value: 0x00000000	
Bit	Name	Description	Access	





## Ethernet Switch on Configurable Logic

2*n+1..2*n	WEIGHT	Port n weight 00 = weight 1 01 = weight 2 10 = weight 3 11 = weight 4  0 <= n <= 15, weight means that a port is polled 1,2,3 or 4 times during one cycle of the round robin process. The round robin process only polls the number of ports that are configured, number of ESOC ports is determined at build time, check register ESOC Config[7..0]	RW
------------	--------	--	----

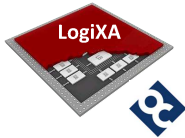
### 4.5 Registers for Search Bus Arbiter

#### 4.5.1 SB\_ARB\_PORT\_DISABLE

Register SB_ARB_PORT_DISABLE		Base Address: 0x8808	
		Offset: 0x7	
		Reset value: 0x00000000	
Bit	Name	Description	Access
n..0	DISABLE	Port n enable/disable internal search bus access 0 = enable 1 = disable  0 <= n <= 15, port enable/disable control for internal search bus access. For test purposes only, use the Ethernet MAC TX and RX enable/disable control bits for enabling and disabling a port during operational use.	RW

#### 4.5.2 SB\_ARB\_PORT\_WEIGHT

Register SB_ARB_PORT_WEIGHT		Base Address: 0x8808	
		Offset: 0x0	
		Reset value: 0x00000000	
Bit	Name	Description	Access



## Ethernet Switch on Configurable Logic

2*n+1..2*n	WEIGHT	Port n weight 00 = weight 1 01 = weight 2 10 = weight 3 11 = weight 4  0 <= n <= 15, weight means that a port is polled 1,2,3 or 4 times during one cycle of the round robin process. The round robin process only polls the number of ports that are configured, number of ESOC ports is determined at build time, check register ESOC Config[7..0]	RW
------------	--------	--	----

### 4.6 Registers for Search Engine

#### 4.6.1 SEARCH\_ENGINE\_SA\_OVERLOAD\_COUNT

Register		SEARCH_ENGINE_SA_OVERLOAD_COUNT	Base Address: 0x8810 Offset: 0x2 Reset value: 0x00000000
Bit	Name	Description	Access
31..0	SA_OVERLOAD_COUNT	Counter represents the number of dropped source MAC addresses due to overload SA buffer	R

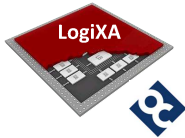
#### 4.6.2 SEARCH\_ENGINE\_SA\_DROP\_COUNT

Register		SEARCH_ENGINE_SA_DROP_COUNT	Base Address: 0x8810 Offset: 0x1 Reset value: 0x00000000
Bit	Name	Description	Access
31..0	SA_DROP_COUNT	Counter represents the number of dropped source MAC addresses due to no space within collision buffer	R

#### 4.6.3 SEARCH\_ENGINE\_STAT\_CTRL

Register		SEARCH_ENGINE_STAT_CTRL	Base Address: 0x8810 Offset: 0x0 Reset value: 0x8000012C*
Bit	Name	Description	Access
31	AGE_TIMER_ENA	Enable AGING mechanism 0 = disabled 1 = enabled	RW
30..12		Not used	
11..0	AGE_TIMER	1x AGE_TIMER seconds < Aging time < 2x AGE_TIMER seconds	RW

\*Reset value depends ESOC\_MODE parameter in ESoCL configuration file, see chapter 2.1.



### 4.7 Registers for Test Bench

#### 4.7.1 RGMII\_PORT\_ENABLE

Register		RGMII_PORT_ENABLE		Base Address: 0xFF00 Offset: 0x0 Reset value: 0x00000000	
Bit	Name	Description	Access		
n	RGMII_PORT_ENABLE	RGMII Port n of test bench control 0 = disable 1 = enable	RW		

## 5. SIMULATION ENVIRONMENT

The simulation environment is semi-automatic and is under control of Modelsim macro's and input files for stimuli on the ESoCL Control interface and the ESoCL Ethernet ports. This chapter describes the simulation environment, how to run a simulation and provides an overview of all available scripts.

### 5.1 Block diagram

The block diagram shows the set-up of the test bench.

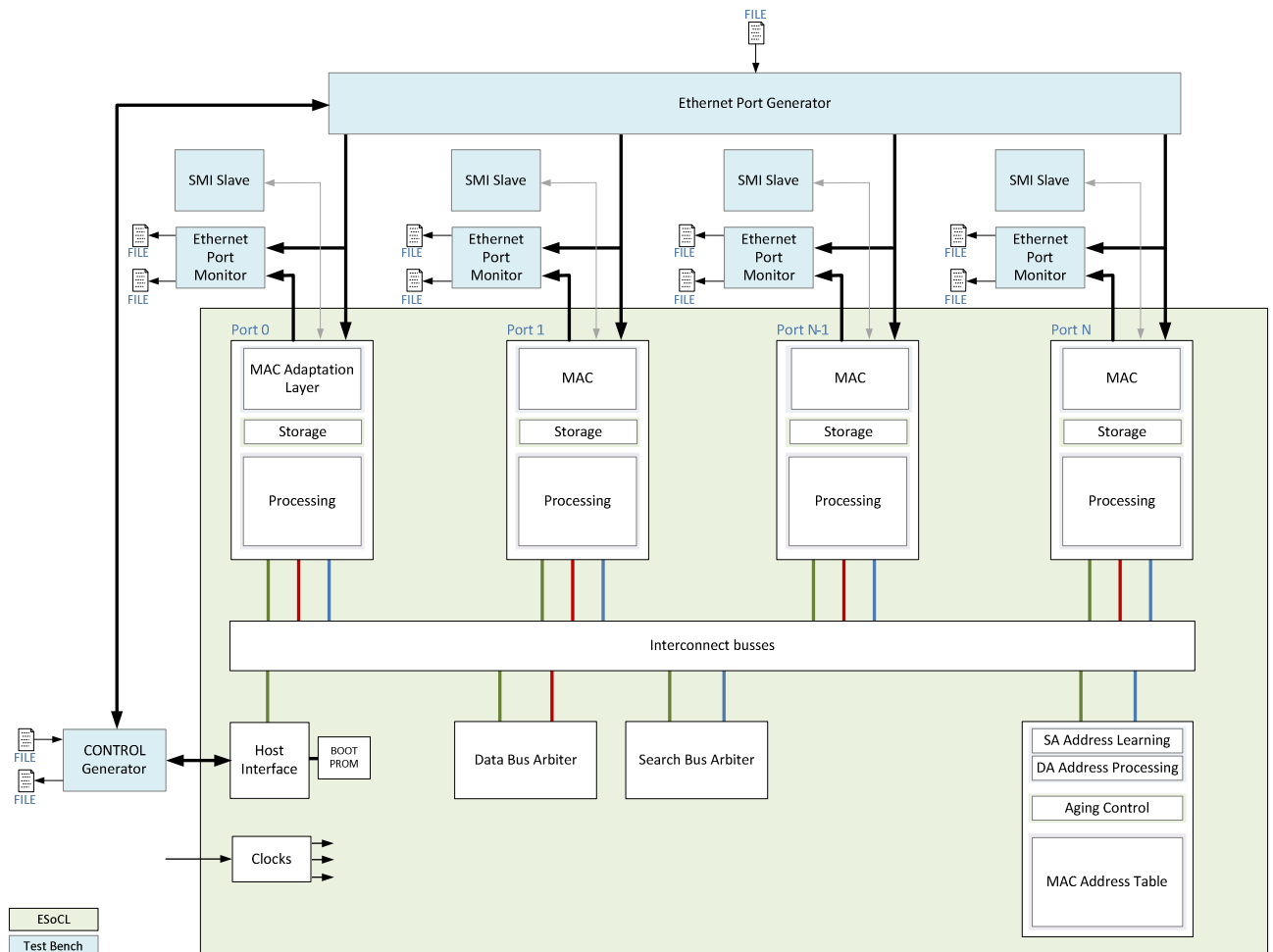
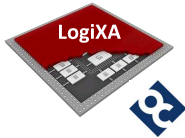


Figure 2: block diagram of test bench










The ESoCL is surrounded by the following units:

- Control Generator, the Control Generator is used to read and write all registers inside the ESoCL. The read and write instructions are retrieved from a file. The result of read and write instructions are logged in a file.



- Ethernet Port Generator, the Ethernet Port Generator is used to send Ethernet packets to an Ethernet port. The Ethernet packets are retrieved from a file. There is one file that can contain packets for all available ports. The Control Generator can start and stop each interface between the Ethernet Port Generator and the ESoCL.
- Ethernet Port Monitor, the Ethernet Port Monitor is used to log all Ethernet packets seen on the input and output of an Ethernet port in a file.
- SMI Slave, the SMI slave is used to validate the SMI (MDIO, MDC) interface of each Ethernet port.

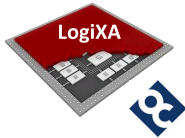
### 5.2 Folder structure

	<drive>:\<folder>\Simulation	Top folder of simulation environment
	\Logs	Log files from Control Generator and Ethernet Port Generator including the used input script files for the Control Generator and Ethernet Port Generator and wave files, each script has its own sub folder
	\0 ... N	
	\Memory_init	Memory initialization files are used to initialize e.g. MAC learning tables before a simulation with pre-filled memories is executed
	\Modelsim	Modelsim simulation folder includes the Modelsim macro's, these macro's are described in chapter 5.3
	\References	Reference log files from Control Generator and Ethernet Port Generator including the used input script files for the Control Generator and Ethernet Port Generator and wave files, each script has its own sub folder. Reference log files are approved log files. A quick folder compare-by-content between the /Logs folder and the /Reference folder shows the differences after a new simulation cycle is executed
	\0 ... N	
	\Scripts	Script files to control the Control Generator and the Ethernet Port Generator.
	\Waves	Wave files used by Modelsim

### 5.3 Macro's

The following Modelsim macro's are available in the folder *drive>:\<folder>\Simulation\Modelsim*:

Macro	Command in Modelsim transcript window	Description
auto.do	do auto.do.	Macro to run all available scripts automatically. After all scripts are executed the log files are



		updated and a compare-by-content with the references can be used to see the differences.
build.do	do build.do	Macro to build the ESoCL into the Modelsim work library
init.do	do init.do <environment>	Macro to initialize your simulation environment, see chapter 2.3 for information about the <environment>.
run.do	do run.do <simulation time in us> <wave file> <Memory init file> <script number>	Macro to run a simulation, see chapter 5.5 for information about available scripts, these script descriptions include the minimum required simulation time, the applicable wave file, the required memory initialization file and the script number.

Table 4: Modelsim macro's

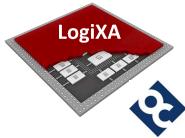
### 5.4 Execute a simulation

Execute the following steps to run a simulation:

- Start Modelsim, see the release bulletin to check the tooling used for the released ESoCL version.
- Go to File → Change Directory ... in Modelsim and browse to the simulation folder <drive>:\<folder>\Simulation\Modelsim
- Initialize your simulation environment by executing the init.do macro, see chapter 5.3 for information about the macro.
- Build the ESoCL by executing the build.do macro, see chapter 5.3 for information about the macro.
- Run a script or all scripts by executing respectively the run.do macro or the auto.do macro, see chapter 5.3 for information about these macro's.

### 5.5 Scripts

All available scripts are executed on the following ESoCL configuration:



- Number of ports: 8
- Bandwidth of a port: 1000Mbps
- Duplex mode of a port: full
- Default VLAN ID of a port: 1

See the corresponding CONTROL generator and Ethernet Port Generator input files in the folder <drive>:\<folder>\Simulation\Scripts for all details of an simulation.

### 5.5.1 Scripts for Initialization

Script:	0	Simulation time	110 us	memory Init:	0	Wave file:	0
Title	Initialization - Read initial register values						
Description	The purpose of this simulation is to check the initial contents after power-up of the design. Most of the registers are initialised according to the register description, but some status registers can deviate from this register description like the PLL Lock status bits, the number of ports in the register CTRL_STAT_CTRL and the version/release number in the register CTRL_VERSION.						

*(Modelsim macro > do run.do 110 0 0 0)*

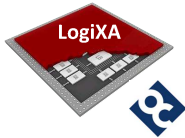
Script:	1	Simulation time	110 us	memory Init:	0	Wave file:	0
Title	Initialization - Configure entities and read register values						
Description	The purpose of this simulation is to check the read/write access to all registers and tables of the ESOC.						

*(Modelsim macro > do run.do 110 0 0 1)*

### 5.5.2 Scripts for basic packet flow

Script:	2	Simulation time	110 us	memory Init:	0	Wave file:	0
Title	Basic packet flow - Packet flow from port to port, untagged packets						
Description	<p>The purpose of this simulation is to check the basic packet forward process of two unicast, untagged packets send from host A to host B and vice versa. Both ports will tag untagged packets with the same port default VLAN ID and both ports are member of this port default VLAN ID.</p> <p>The first untagged packet - associated with the port default VLAN ID - from host A to host B will be broadcasted to all ports, because no addresses are learned by the learn process and all ports are member of the associated VLAN ID. The learn process shall learn the source address of this packet from host A together with this associated VLAN ID.</p> <p>The first untagged packet - associated with the same port default VLAN ID - from host B to host A will not be broadcasted because the learn process already learned the source address and its associated VLAN ID of host A. The learn process shall learn the source address of this packet from host B together with this associated VLAN ID.</p> <p>The second untagged packet - associated with the same port default VLAN ID - from host A and host B is directly forwarded, because destination is known.</p>						

*(Modelsim macro > do run.do 110 0 0 2)*



## Ethernet Switch on Configurable Logic

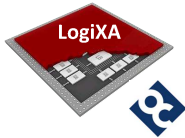
Script:	3	Simulation time	110 us	memory Init:	0	Wave file:	0
Title	Basic packet flow - Packet flow from port to port, tagged packets						
Description	<p>The purpose of this simulation is to check the basic packet forward process of two unicast, tagged packets send from host A to host B and vice versa. Both ports are member of this port default VLAN ID.</p> <p>The first tagged packet from host A to host B will be broadcasted to all ports, because no addresses are learned by the learn process and all ports are member of the associated VLAN ID. The learn process shall learn the source address of this packet from host A together with this associated VLAN ID.</p> <p>The first tagged packet from host B to host A will not be broadcasted because the learn process already learned the source address and its associated VLAN ID of host A. The learn process shall learn the source address of this packet from host B together with this associated VLAN ID.</p> <p>The second tagged packet from host A and host B is directly forwarded, because destination is known.</p>						

(Modelsim macro > do run.do 110 0 0 3)

Script:	4	Simulation time	110 us	memory Init:	0	Wave file:	0
Title	Basic packet flow - Packet flow from port to port, untagged and tagged packets						
Description	<p>The purpose of this simulation is to check the basic packet forward process of two unicast, untagged and tagged packets send from host A to host B and vice versa. Both ports will tag untagged packets with the same port default VLAN ID and both ports are member of this port default VLAN ID.</p> <p>The first untagged packet - associated with the port default VLAN ID - from host A to host B will be broadcasted to all ports, because no addresses are learned by the learn process and all ports are member of the associated VLAN ID. The learn process shall learn the source address of this packet from host A together with this associated VLAN ID.</p> <p>The second tagged packet from host A to host B will be broadcasted to all ports, because no addresses are learned by the learn process and all ports are member of the associated VLAN ID. The learn process shall learn the source address of this packet from host A together with this associated VLAN ID.</p> <p>The first untagged packet - associated with the same port default VLAN ID - from host B to host A will not be broadcasted because the learn process already learned the source address and its associated VLAN ID of host A. The learn process shall learn the source address of this packet from host B together with this associated VLAN ID.</p> <p>The second tagged packet from host B to host A will not be broadcasted because the learn process already learned the source address and its associated VLAN ID of host A. The learn process shall learn the source address of this packet from host B together with this associated VLAN ID.</p>						

(Modelsim macro > do run.do 110 0 0 4)



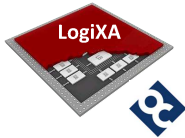


Script:	5	Simulation time	120 us	memory Init:	0	Wave file:	0
Title	Basic packet flow - Packet flow from port to port, tagged packets						
Description	<p>The purpose of this simulation is to check the basic packet forward process of two unicast, tagged packets send from host A to host B and vice versa and two unicast, tagged packets send from host C to host D and vice versa. Host A and B are in a different VLAN than host C and D.</p> <p>The first tagged packet from host A to host B will be broadcasted to some ports, because no addresses are learned by the learn process and only a few ports are member of the associated VLAN ID. The learn process shall learn the source address of this packet from host A together with this associated VLAN ID.</p> <p>The first tagged packet from host B to host A will not be broadcasted because the learn process already learned the source address and its associated VLAN ID of host A. The learn process shall learn the source address of this packet from host B together with this associated VLAN ID.</p> <p>The second tagged packet from host A and host B is directly forwarded, because destination is known.</p> <p>The first tagged packet from host C to host D will be broadcasted to some ports, because no addresses are learned by the learn process and only a few ports are member of the associated VLAN ID. The learn process shall learn the source address of this packet from host C together with this associated VLAN ID.</p> <p>The first tagged packet from host D to host C will not be broadcasted because the learn process already learned the source address and its associated VLAN ID of host C. The learn process shall learn the source address of this packet from host D together with this associated VLAN ID.</p> <p>The second tagged packet from host C and host D is directly forwarded, because destination is known.</p>						

(Modelsim macro > do run.do 120 0 0 5)

Script:	6	Simulation time	110 us	memory Init:	0	Wave file:	0
Title	Basic packet flow - Packet flow for tagged multicast and broadcast packets						
Description	<p>The purpose of this simulation is to check the basic packet forward process of two multicast/broadcast, tagged packets send from host A and host B. All ports are member of the same VLAN IDs.</p> <p>The first and second tagged packet from host A will be broadcasted to all ports, because multicast or broadcast packets are always forwarded to all ports that are member of the VLAN ID.</p> <p>The first and second tagged packet from host B will be broadcasted to all ports, because multicast or broadcast packets are always forwarded to all ports that are member of the VLAN ID.</p>						

(Modelsim macro > do run.do 110 0 0 6)



<b>Script:</b>	7	Simulation time	110 us	memory Init:	0	Wave file:	0
<b>Title</b>	Basic packet flow - Packet flow for tagged multicast and broadcast packets						
<b>Description</b>	<p>The purpose of this simulation is to check the basic packet forward process of two multicast/broadcast, tagged packets send from host A and host B. Not all ports are member of the same VLAN IDs.</p> <p>The first and second tagged packet from host A will be broadcasted to some ports, because multicast or broadcast packets are always forwarded to all ports that are member of the VLAN ID.</p> <p>The first and second tagged packet from host B will be broadcasted to some ports, because multicast or broadcast packets are always forwarded to all ports that are member of the VLAN ID.</p>						

(Modelsim macro > do run.do 110 0 0 7)

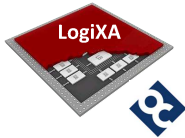
<b>Script:</b>	8	Simulation time	110 us	memory Init:	0	Wave file:	0
<b>Title</b>	Basic packet flow - Packet flow for untagged multicast and broadcast packets						
<b>Description</b>	<p>The purpose of this simulation is to check the basic packet forward process of two multicast/broadcast, untagged packets send from host A and host B. Not all ports have the same port default VLAN ID.</p> <p>The first and second untagged packet - associated with the port default VLAN ID - from host A will be broadcasted to some ports, because multicast or broadcast packets are always forwarded to all ports that are member of the VLAN ID.</p> <p>The first and second untagged packet - associated with the port default VLAN ID - from host B will be broadcasted to some ports, because multicast or broadcast packets are always forwarded to all ports that are member of the VLAN ID.</p>						

(Modelsim macro > do run.do 110 0 0 8)

<b>Script:</b>	9	Simulation time	110 us	memory Init:	0	Wave file:	0
<b>Title</b>	Basic packet flow - Packet flow for untagged multicast and broadcast packets						
<b>Description</b>	<p>The purpose of this simulation is to check the basic packet forward process of two multicast/broadcast, untagged packets send from host A and host B. All ports have the same port default VLAN ID.</p> <p>The first and second untagged packet - associated with the port default VLAN ID - from host A will be broadcasted to all ports, because multicast or broadcast packets are always forwarded to all ports that are member of the VLAN ID.</p> <p>The first and second untagged packet - associated with the port default VLAN ID - from host B will be broadcasted to all ports, because multicast or broadcast packets are always forwarded to all ports that are member of the VLAN ID.</p>						

(Modelsim macro > do run.do 110 0 0 9)

<b>Script:</b>	10	Simulation time	110 us	memory Init:	0	Wave file:	0
----------------	----	-----------------	--------	--------------	---	------------	---



Title	Basic packet flow - Packet flow for untagged packets from multiple ports to one port
Description	<p>The purpose of this simulation is to check the basic packet forward process of multiple unicast, untagged packets from host A...G to host H. All ports will tag untagged packets with the same port default VLAN ID and all ports are member of this port default VLAN ID.</p> <p>The first untagged packet - associated with the port default VLAN ID - from host H to host A will be broadcasted to all ports, because no addresses are learned by the learn process and all ports are member of the associated VLAN ID. The learn process shall learn the source address of this packet from host H together with this associated VLAN ID.</p> <p>All untagged packets - associated with the same port default VLAN ID - from host A...G to host H will not be broadcasted because the learn process already learned the source address and its associated VLAN ID of host H. The learn process shall learn the source addresses of the packets from host A...G together with this associated VLAN ID.</p>

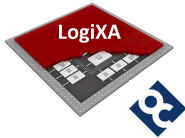
(Modelsim macro > do run.do 110 0 0 10)

Script:	11	Simulation time	150 us	memory Init:	0	Wave file:	0
Title	Basic packet flow - Packet flow for untagged packets from multiple ports to one port						
Description	<p>The purpose of this simulation is to check the basic packet forward process of multiple unicast, untagged packets from host A...G to host H. All ports will tag untagged packets with the same port default VLAN ID and all ports are member of this port default VLAN ID. The size of the packets from host A is set to maximum, while the packets from host B...G are set to minimum.</p> <p>The first untagged packet - associated with the port default VLAN ID - from host H to host A will be broadcasted to all ports, because no addresses are learned by the learn process and all ports are member of the associated VLAN ID. The learn process shall learn the source address of this packet from host H together with this associated VLAN ID.</p> <p>The first large untagged packet - associated with the same port default VLAN ID - from host A to host H will not be broadcasted because the learn process already learned the source address and its associated VLAN ID of host H. The learn process shall learn the source addresses of the packet from host A together with this associated VLAN ID. During the transfer of this large packet the small packets from host B...G are processed, which means the destination address inside the small packet is provided to the search engine and the search engine will reply with the destination port. All packets of host B...G are processed during the transfer of the large packet from host A and will be transferred to host H when the large packet is transferred.</p> <p>All small untagged packets - associated with the same port default VLAN ID - from host B...G to host H will not be broadcasted because the learn process already learned the source address and its associated VLAN ID of host H. The learn process shall learn the source addresses of the packets from host B...G together with this associated VLAN ID. These small packets from hosts B...G are send one by one because all ports have the same priority on the internal data bus.</p>						

(Modelsim macro > do run.do 150 0 0 11)

### 5.5.3 Scripts for prioritized packet flow

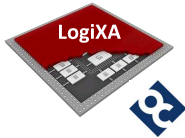
Script:	12	Simulation time	150 us	memory Init:	0	Wave file:	0
---------	----	-----------------	--------	--------------	---	------------	---



Title	Prioritized packet flow - Packet flow for untagged packets from multiple ports to one port
Description	<p>The purpose of this simulation is to check the basic packet forward process of multiple unicast, untagged packets from host A...G to host H. All ports will tag untagged packets with the same port default VLAN ID and all ports are member of this port default VLAN ID. The size of the packets from host A is set to maximum, while the packets from host B...G are set to minimum. The priority of the ports connected to host B to G on the internal data bus of the ESOC is set to 2, instead of the default value of 1.</p> <p>The first untagged packet - associated with the port default VLAN ID - from host H to host A will be broadcasted to all ports, because no addresses are learned by the learn process and all ports are member of the associated VLAN ID. The learn process shall learn the source address of this packet from host H together with this associated VLAN ID.</p> <p>The first large untagged packet - associated with the same port default VLAN ID - from host A to host H will not be broadcasted because the learn process already learned the source address and its associated VLAN ID of host H. The learn process shall learn the source addresses of the packet from host A together with this associated VLAN ID. During the transfer of this large packet the small packets from host B...G are processed, which means the destination address inside the small packet is provided to the search engine and the search engine will reply with the destination port. All packets of host B...G are processed during the transfer of the large packet from host A and will be transferred to host H when the large packet is transferred.</p> <p>All small untagged packets - associated with the same port default VLAN ID - from host B...G to host H will not be broadcasted because the learn process already learned the source address and its associated VLAN ID of host H. The learn process shall learn the source addresses of the packets from host B...G together with this associated VLAN ID. These small packets from hosts B...G are send not one by one per port because these ports have various priority setting - not the default of 1 - on the internal data bus.</p>

*(Modelsim macro > do run.do 150 0 0 12)*

Script:	13	Simulation time	150 us	memory Init:	0	Wave file:	0
Title	Prioritized packet flow - Packet flow for untagged packets from multiple ports to one port						
Description	<p>The purpose of this simulation is to check the basic packet forward process of multiple unicast, untagged packets from host A...G to host H. All ports will tag untagged packets with the same port default VLAN ID and all ports are member of this port default VLAN ID. The size of the packets from host A and B is set to maximum, while the packets from host C...G are set to minimum. The priority of the ports connected to host C to G on the internal data bus of the ESOC is set to various values, instead of the default value of 1.</p> <p>The first untagged packet - associated with the port default VLAN ID - from host H to host A will be broadcasted to all ports, because no addresses are learned by the learn process and all ports are member of the associated VLAN ID. The learn process shall learn the source address of this packet from host H together with this associated VLAN ID.</p> <p>The first large untagged packet - associated with the same port default VLAN ID - from host A to host H will not be broadcasted because the learn process already learned the source address and its associated VLAN ID of host H. The learn process shall learn the source addresses of the packet from host A together with this associated VLAN ID. The second large untagged packet - associated with the same port default VLAN ID - from host B to host H will not be broadcasted because the learn process already learned the source address and its associated VLAN ID of host H. The learn process shall learn the source addresses of the</p>						



	<p>packet from host B together with this associated VLAN ID. During the transfer of these large packets the small packets from host C...G are processed, which means the destination address inside the small packet is provided to the search engine and the search engine will reply with the destination port. All packets of host C...G are processed during the transfer of the large packets from host A and B and will be transferred to host H when the large packets are transferred.</p> <p>All small untagged packets - associated with the same port default VLAN ID - from host C...G to host H will not be broadcasted because the learn process already learned the source address and its associated VLAN ID of host H. The learn process shall learn the source addresses of the packets from host C...G together with this associated VLAN ID. These small packets from hosts C...G are send not one by one per port because these ports have various priority setting - not the default of 1 - on the internal data bus.</p>
--	---

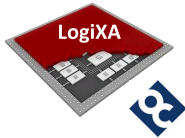
*(Modelsim macro > do run.do 150 0 0 13)*

### 5.5.4 Scripts for Output congestion control

Script:	14	Simulation time	500 us	memory Init:	0	Wave file:	0
Title	Output congestion control - Packet flow for untagged packets from multiple ports to one overloaded port						
Description	<p>The purpose of this simulation is to check the behaviour of the switch when a destination port is congested.</p> <p>The first untagged packet - associated with the port default VLAN ID - from host H to host A will be broadcasted to all ports, because no addresses are learned by the learn process and all ports are member of the associated VLAN ID. The learn process shall learn the source address of this packet from host H together with this associated VLAN ID.</p> <p>Then sixteen large untagged packets - associated with the same port default VLAN ID - from host A to host H will not be broadcasted because the learn process already learned the source address and its associated VLAN ID of host H. The learn process shall learn the source addresses of the packet from host A together with this associated VLAN ID. The eleventh packet from host A to host H is dropped by port 7 of the switch, the FIFO in the MAL layer plus the FIFO in the MAC layer provides a total FIFO space of 10.4 maximum sized (1518 bytes) untagged packets.</p> <p>The congested port 7 indicates towards the search engine his status. The following 5 packets from host A to host H are dropped at port 0, because the search requests for these packets results in destination port 7, but because this port is congested the search engine returns the null result.</p>						

*(Modelsim macro > do run.do 500 0 0 14)*

Script:	15	Simulation time	500 us	memory Init:	0	Wave file:	0
Title	Output congestion control - Packet flow for untagged packets from multiple ports to one overloaded port						
Description	<p>The purpose of this simulation is to check the behaviour of the switch when a destination port is congested.</p> <p>The first untagged packet - associated with the port default VLAN ID - from host H to host A will be broadcasted to all ports, because no addresses are learned by the learn process and all ports are member of the associated VLAN ID. The learn process shall learn the source</p>						



## Ethernet Switch on Configurable Logic

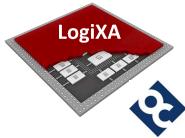
	<p>address of this packet from host H together with this associated VLAN ID.</p> <p>Then sixteen large untagged packets - associated with the same port default VLAN ID - from host A...G to host H will not be broadcasted because the learn process already learned the source address and its associated VLAN ID of host H. The learn process shall learn the source addresses of the packet from host A...G together with this associated VLAN ID. The eleventh packet to host H is dropped by port 7 of the switch, the FIFO in the MAL layer plus the FIFO in the MAC layer provides a total FIFO space of 10.4 maximum sized (1518 bytes) untagged packets.</p> <p>The congested port 7 indicates towards the search engine his status. The following 5 packets from host A...G to host H are dropped at port 0...6, because the search requests for these packets results in destination port 7, but because this port is congested the search engine returns the null result.</p>
--	--

*(Modelsim macro > do run.do 500 0 0 15)*

<b>Script:</b>	<b>16 Simulation time 500 us memory Init: 0 Wave file: 0</b>
<b>Title</b>	Output congestion control - Packet flow for untagged packets from multiple ports to one overloaded port
<b>Description</b>	<p>The purpose of this simulation is to check the behaviour of the switch when a destination port is congested.</p> <p>The first untagged packet - associated with the port default VLAN ID - from host H to host A will be broadcasted to all ports, because no addresses are learned by the learn process and all ports are member of the associated VLAN ID. The learn process shall learn the source address of this packet from host H together with this associated VLAN ID.</p> <p>Then sixteen large untagged packets - associated with the same port default VLAN ID - from host A to host H will not be broadcasted because the learn process already learned the source address and its associated VLAN ID of host H. The learn process shall learn the source addresses of the packet from host A together with this associated VLAN ID. The eleventh packet from host A to host H is dropped by port 7 of the switch, the FIFO in the MAL layer plus the FIFO in the MAC layer provides a total FIFO space of 10.4 maximum sized (1518 bytes) untagged packets.</p> <p>The congested port 7 indicates towards the search engine his status. The following 5 packets from host A to host H are dropped at port 0, because the search requests for these packets results in destination port 7, but because this port is congested the search engine returns the null result.</p> <p>Four large untagged packets - associated with the same port default VLAN ID - will be send from host A to host H after the congestion at port 7 is resolved!</p>

*(Modelsim macro > do run.do 500 0 0 16)*

<b>Script:</b>	<b>17 Simulation time 550 us memory Init: 0 Wave file: 0</b>
<b>Title</b>	Output congestion control - Packet flow for untagged packets from multiple ports to one overloaded port
<b>Description</b>	The purpose of this simulation is to check the behaviour of the switch when a destination



	<p>port is congested.</p> <p>The first untagged packet - associated with the port default VLAN ID - from host H to host A will be broadcasted to all ports, because no addresses are learned by the learn process and all ports are member of the associated VLAN ID. The learn process shall learn the source address of this packet from host H together with this associated VLAN ID.</p> <p>Then sixteen large untagged packets - associated with the same port default VLAN ID - from host A...G to host H will not be broadcasted because the learn process already learned the source address and its associated VLAN ID of host H. The learn process shall learn the source addresses of the packet from host A...G together with this associated VLAN ID. The eleventh packet to host H is dropped by port 7 of the switch, the FIFO in the MAL layer plus the FIFO in the MAC layer provides a total FIFO space of 10.4 maximum sized (1518 bytes) untagged packets.</p> <p>The congested port 7 indicates towards the search engine his status. The following 5 packets from host A...G to host H are dropped at port 0...6, because the search requests for these packets results in destination port 7, but because this port is congested the search engine returns the null result.</p> <p>Four large untagged packets - associated with the same port default VLAN ID - will be send from host A...G to host H after the congestion at port 7 is resolved!</p>
--	--

*(Modelsim macro > do run.do 550 0 0 17)*

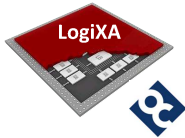
### 5.5.5 Scripts for Input congestion control

<b>Script:</b>	<b>18</b>	<b>Simulation time</b>	<b>350 us</b>	<b>memory Init:</b>	<b>0</b>	<b>Wave file:</b>	<b>0</b>
<b>Title</b>	Input congestion control - Packet overflow in MAL FIFO						
<b>Description</b>	<p>The purpose of this simulation is to check the behaviour of the switch when a source port is congested.</p> <p>The first untagged packet - associated with the port default VLAN ID - from host H to host A will be broadcasted to all ports, because no addresses are learned by the learn process and all ports are member of the associated VLAN ID. The learn process shall learn the source address of this packet from host H together with this associated VLAN ID.</p> <p>Then ten large untagged packets - associated with the same port default VLAN ID - from host A to host H will not be transferred, because the inbound port is blocked (test feature of the data bus arbiter). The blocked port results in a FIFO Full situation of the MAL function, the port is unblocked before the MAC FIFO becomes full. Simultaneously one large untagged packet - associated with the same port default VLAN ID - from host B...G to host H will not be broadcasted because the learn process already learned the source address and its associated VLAN ID of host H. The learn process shall learn the source addresses of the packet from host A...G together with this associated VLAN ID.</p> <p>At the end host H will receive ten packets from host A and one packet from host B...G.</p>						

*(Modelsim macro > do run.do 350 0 0 18)*

<b>Script:</b>	<b>19</b>	<b>Simulation time</b>	<b>450 us</b>	<b>memory Init:</b>	<b>0</b>	<b>Wave file:</b>	<b>0</b>
<b>Title</b>	Input congestion control - Packet overflow in MAL and MAC FIFO						
<b>Description</b>	The purpose of this simulation is to check the behaviour of the switch when a source port is						





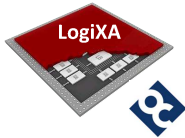
	<p>congested.</p> <p>The first untagged packet - associated with the port default VLAN ID - from host H to host A will be broadcasted to all ports, because no addresses are learned by the learn process and all ports are member of the associated VLAN ID. The learn process shall learn the source address of this packet from host H together with this associated VLAN ID.</p> <p>Then thirteen large untagged packets - associated with the same port default VLAN ID - from host A to host H will not be transferred, because the inbound port is blocked (test feature of the data bus arbiter). The blocked port results in a FIFO Full situation of the MAL function and the port is not unblocked before the MAC FIFO becomes full, packets will be dropped. Simultaneously one large untagged packet - associated with the same port default VLAN ID - from host B...G to host H will not be broadcasted because the learn process already learned the source address and its associated VLAN ID of host H. The learn process shall learn the source addresses of the packet from host A...G together with this associated VLAN ID. After a while the blocked port is unblocked and three small packets - associated with the same port default VLAN ID - from host A to host H will not be broadcasted because the learn process already learned the source address and its associated VLAN ID of host H.</p> <p>At the end host H will receive thirteen packets from host A and one packet from host B...G.</p>
--	--

*(Modelsim macro > do run.do 450 0 0 19)*

### 5.5.6 Scripts for search engine SA

Script:	20 Simulation time 1000 us memory Init: 0 Wave file: 0
Title	Search engine SA - Source Address learning and aging, first hit in collision buffer
Description	<p>The purpose of this simulation is to check the behaviour of the search engine and the aging mechanism</p> <p>The first untagged packet - associated with the port default VLAN ID - from host A to host H will be broadcasted to all ports, because no addresses are learned by the learn process and all ports are member of the associated VLAN ID. The learn process shall learn the source address of this packet from host A together with this associated VLAN ID and shall store the address at the first entry of the pointed collision buffer, the update and valid flag of the entry will be set.</p> <p>The second untagged packet - associated with the port default VLAN ID - from host H to host A will be unicasted to the destination port, because the address and VLAN ID of host A is found in the MAC Address learning table. The learn process shall learn the source address of this packet from host H together with this associated VLAN ID and shall store the address at the first entry of the pointed collision buffer, the update and valid flag of the entry will be set.</p> <p>The third and fourth untagged packet - associated with the port default VLAN ID - from host A to host H and vice versa will be unicasted to the destination port, because the addresses and VLAN IDs of host A and host H are found in the MAC Address learning table, but the update flag was reset by the aging process. The learn process shall set the update again because the addresses are seen within the age time.</p> <p>The fifth untagged packet - associated with the port default VLAN ID - from host A to host H will be broadcasted to all ports, because the current entries in the MAC Learning table related to host A and G are out of age - VALID and UPDATE flag reset - and so the destination</p>



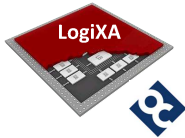


address and the associated VLAN ID are unknown. The learn process shall re-learn the source address of this packet from host A together with this associated VLAN ID and shall store the address at the first entry of the pointed collision buffer, the update and valid flag of the entry will be set.

The sixth untagged packet - associated with the port default VLAN ID - from host H to host A will be unicasted to the destination port, because the address and VLAN ID of host A is found in the MAC Address learn table. The current entry in the MAC Learning table related to host G is out of age. The learn process shall re-learn the source address of this packet from host H together with this associated VLAN ID and shall store the address at the first entry of the pointed collision buffer, the update and valid flag of the entry will be set.

*(Modelsim macro > do run.do 1000 0 0 20)*

Script:	21 Simulation time 1000 us memory Init: 1 Wave file: 0
Title	Search engine SA - Source Address learning and aging, middle hit in collision buffer
Description	<p>The purpose of this simulation is to check the behaviour of the search engine and the aging mechanism</p> <p>The first untagged packet - associated with the port default VLAN ID - from host A to host H will be broadcasted to all ports, because no addresses are learned by the learn process and all ports are member of the associated VLAN ID. The learn process shall learn the source address of this packet from host A together with this associated VLAN ID and shall store the address at the middle entry of the pointed collision buffer, the update and valid flag of the entry will be set.</p> <p>The second untagged packet - associated with the port default VLAN ID - from host H to host A will be unicasted to the destination port, because the address and VLAN ID of host A is found in the MAC Address learning table. The learn process shall learn the source address of this packet from host H together with this associated VLAN ID and shall store the address at the middle entry of the pointed collision buffer, the update and valid flag of the entry will be set.</p> <p>The third and fourth untagged packet - associated with the port default VLAN ID - from host A to host H and vice versa will be unicasted to the destination port, because the addresses and VLAN IDs of host A and host H are found in the MAC Address learning table, but the update flag was reset by the aging process. The learn process shall set the update again because the addresses are seen within the age time.</p> <p>The fifth untagged packet - associated with the port default VLAN ID - from host A to host H will be broadcasted to all ports, because the current entries in the MAC Learning table related to the collision buffer of host A and G are out of age - VALID and UPDATE flag reset - and so the destination address and the associated VLAN ID are unknown. The learn process shall re-learn the source address of this packet from host A together with this associated VLAN ID and shall store the address at the first entry of the pointed collision buffer, the update and valid flag of the entry will be set.</p> <p>The sixth untagged packet - associated with the port default VLAN ID - from host H to host A will be unicasted to the destination port, because the address and VLAN ID of host A is found in the MAC Address learn table. The current entry in the MAC Learning table related to the collision buffer of host G are out of age. The learn process shall re-learn the source address of this packet from host H together with this associated VLAN ID and shall store the address at the first entry of the pointed collision buffer, the update and valid flag of the entry will be</p>



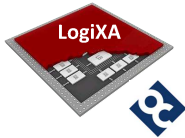
set.

*(Modelsim macro > do run.do 1000 0 1 21)*

Script:	22	Simulation time	1000 us	memory Init:	2	Wave file:	0
Title	Search engine SA - Source Address learning and aging, last hit in collision buffer						
Description	<p>The purpose of this simulation is to check the behaviour of the search engine and the aging mechanism</p> <p>The first untagged packet - associated with the port default VLAN ID - from host A to host H will be broadcasted to all ports, because no addresses are learned by the learn process and all ports are member of the associated VLAN ID. The learn process shall learn the source address of this packet from host A together with this associated VLAN ID and shall store the address at the last entry of the pointed collision buffer, the update and valid flag of the entry will be set.</p> <p>The second untagged packet - associated with the port default VLAN ID - from host H to host A will be unicasted to the destination port, because the address and VLAN ID of host A is found in the MAC Address learning table. The learn process shall learn the source address of this packet from host H together with this associated VLAN ID and shall store the address at the last entry of the pointed collision buffer, the update and valid flag of the entry will be set.</p> <p>The third and fourth untagged packet - associated with the port default VLAN ID - from host A to host H and vice versa will be unicasted to the destination port, because the addresses and VLAN IDs of host A and host H are found in the MAC Address learning table, but the update flag was reset by the aging process. The learn process shall set the update again because the addresses are seen within the age time.</p> <p>The fifth untagged packet - associated with the port default VLAN ID - from host A to host H will be broadcasted to all ports, because the current entries in the MAC Learning table related to the collision buffer of host A and G are out of age - VALID and UPDATE flag reset - and so the destination address and the associated VLAN ID are unknown. The learn process shall re-learn the source address of this packet from host A together with this associated VLAN ID and shall store the address at the first entry of the pointed collision buffer, the update and valid flag of the entry will be set.</p> <p>The sixth untagged packet - associated with the port default VLAN ID - from host H to host A will be unicasted to the destination port, because the address and VLAN ID of host A is found in the MAC Address learn table. The current entry in the MAC Learning table related to the collision buffer of host G are out of age. The learn process shall re-learn the source address of this packet from host H together with this associated VLAN ID and shall store the address at the first entry of the pointed collision buffer, the update and valid flag of the entry will be set.</p>						

*(Modelsim macro > do run.do 1000 0 2 22)*

Script:	23	Simulation time	1000 us	memory Init:	3	Wave file:	0
Title	Search engine SA - Source Address learning and aging, overrun in collision buffer						
Description	<p>The purpose of this simulation is to check the behaviour of the search engine and the aging mechanism</p>						



	<p>The first untagged packet - associated with the port default VLAN ID - from host A to host H will be broadcasted to all ports, because no addresses are learned by the learn process and all ports are member of the associated VLAN ID. The learn process shall not learn the source address of this packet from host A together with this associated VLAN ID because the pointed collision buffer is full!</p> <p>The second untagged packet - associated with the port default VLAN ID - from host H to host A will be broadcasted to all ports, because no addresses are learned by the learn process and all ports are member of the associated VLAN ID. The learn process shall not learn the source address of this packet from host H together with this associated VLAN ID because the pointed collision buffer is full!</p> <p>The third and fourth untagged packet - associated with the port default VLAN ID - from host H to host A and vice versa will be broadcasted to all ports, because no addresses are learned by the learn process and all ports are member of the associated VLAN ID. The learn process shall not learn the source address of this packet from host A and host H together with this associated VLAN ID because the pointed collision buffer is full!</p> <p>The fifth untagged packet - associated with the port default VLAN ID - from host A to host H will be broadcasted to all ports and the associated VLAN ID are unknown. The learn process shall learn the source address of this packet from host A together with this associated VLAN ID and shall store the address at the first entry of the pointed collision buffer, the update and valid flag of the entry will be set.</p> <p>The sixth untagged packet - associated with the port default VLAN ID - from host H to host A will be unicasted to the destination port, because the address and VLAN ID of host A is found in the MAC Address learn table. The learn process shall learn the source address of this packet from host H together with this associated VLAN ID and shall store the address at the first entry of the pointed collision buffer, the update and valid flag of the entry will be set.</p>
--	--

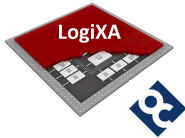
*(Modelsim macro > do run.do 1000 0 3 23)*

### 5.5.7 Scripts for Search engine DA

<b>Script:</b>	24	Simulation time	100 us	memory Init:	4	Wave file:	0
<b>Title</b>	Search engine DA - Destination Address searching, first hit in collision buffer						
<b>Description</b>	<p>The purpose of this simulation is to check the behaviour of the search engine, MAC address learning filled with valid entries.</p> <p>The first untagged packet - associated with the port default VLAN ID - from host A to host H will be unicasted to the destination port, because the address and VLAN ID of host H is found in the MAC Address learning table in the first entry of the pointed collision buffer.</p> <p>The second untagged packet - associated with the port default VLAN ID - from host H to host A will be unicasted to the destination port, because the address and VLAN ID of host A is found in the MAC Address learning table in the first entry of the pointed collision buffer.</p>						

*(Modelsim macro > do run.do 100 0 4 24)*

<b>Script:</b>	25	Simulation time	100 us	memory Init:	5	Wave file:	0
<b>Title</b>	Search engine DA - Destination Address searching, middle hit in collision buffer						
<b>Description</b>	The purpose of this simulation is to check the behaviour of the search engine, MAC address learning filled with valid entries.						



	<p>The first untagged packet - associated with the port default VLAN ID - from host A to host H will be unicasted to the destination port, because the address and VLAN ID of host H is found in the MAC Address learning table in the middle entry of the pointed collision buffer.</p> <p>The second untagged packet - associated with the port default VLAN ID - from host H to host A will be unicasted to the destination port, because the address and VLAN ID of host A is found in the MAC Address learning table in the middle entry of the pointed collision buffer.</p>
--	--

*(Modelsim macro > do run.do 100 0 5 25)*

<b>Script:</b>	<b>26</b> Simulation time 100 us memory Init: 6 Wave file: 0
<b>Title</b>	Search engine DA - Destination Address searching, last hit in collision buffer
<b>Description</b>	<p>The purpose of this simulation is to check the behaviour of the search engine, MAC address learning filled with valid entries.</p> <p>The first untagged packet - associated with the port default VLAN ID - from host A to host H will be unicasted to the destination port, because the address and VLAN ID of host H is found in the MAC Address learning table in the last entry of the pointed collision buffer.</p> <p>The second untagged packet - associated with the port default VLAN ID - from host H to host A will be unicasted to the destination port, because the address and VLAN ID of host A is found in the MAC Address learning table in the last entry of the pointed collision buffer.</p>

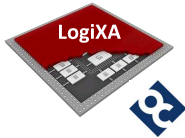
*(Modelsim macro > do run.do 100 0 6 26)*

### 5.5.8 Scripts for VLAN Tag Editing

<b>Script:</b>	<b>27</b> Simulation time 100 us memory Init: 0 Wave file: 0
<b>Title</b>	VLAN Tag Editing - Inbound QoS Tagged packet with VLAN ID 0x000
<b>Description</b>	<p>The purpose of this simulation is to check the processing of a QoS only packet, a VLAN tagged packet with VLAN ID 0.</p> <p>The first untagged packet - associated with the port default VLAN ID - from host A is broadcasted to all ports, the packet header is unmodified. The second tagged packet from host A is broadcasted to all ports, the packet header is modified because it is a QoS Only packet with VLAN ID 0. The tag is replaced with the default VLAN ID of the receiving port. The third tagged packet from host A is broadcasted to all ports, the packet header is unmodified. The fourth untagged packet - associated with the port default VLAN ID - from host A is broadcasted to all ports, the packet header is unmodified.</p>

*(Modelsim macro > do run.do 100 0 0 27)*

<b>Script:</b>	<b>28</b> Simulation time 100 us memory Init: 0 Wave file: 0
<b>Title</b>	VLAN Tag Editing - Inbound tagged packet with VLAN Tag replacement enabled
<b>Description</b>	<p>The purpose of this simulation is to check the processing of a tagged packet when replace VLAN tag by port default VLAN tag is enabled.</p> <p>The first untagged packet - associated with the port default VLAN ID - from host A is</p>



broadcasted to all ports, the packet header is unmodified. The second tagged packet from host A is broadcasted to all ports, the packet header is modified because it is a QoS Only packet with VLAN ID 0. The tag is replaced with the default VLAN ID of the receiving port. The third tagged packet from host A is broadcasted to all ports, the packet header is modified because tag replacement is enabled. The tag is replaced with the default VLAN ID of the receiving port. The fourth untagged packet - associated with the port default VLAN ID - from host A is broadcasted to all ports, the packet header is unmodified.

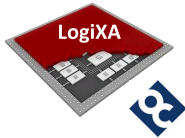
*(Modelsim macro > do run.do 100 0 0 28)*

Script:	29	Simulation time	100 us	memory Init:	0	Wave file:	0
Title	VLAN Tag Editing - Outbound tagged packet with VLAN Tag replacement enabled						
Description	<p>The purpose of this simulation is to check the processing of a tagged packet when replace VLAN tag by port default VLAN tag is enabled.</p> <p>The first untagged packet - associated with the port default VLAN ID - from host A is broadcasted to all ports, the packet header is unmodified. The second tagged packet from host A is broadcasted to all ports, the packet header is unmodified because it has the default VLAN ID. The third tagged packet from host A is broadcasted to all ports, the packet header is modified because tag replacement is enabled. The tag is replaced with the default VLAN ID on port 7, all other ports do not modify the packet header. The fourth untagged packet - associated with the port default VLAN ID - from host A is broadcasted to all ports, the packet header is unmodified.</p>						

*(Modelsim macro > do run.do 100 0 0 29)*

### 5.5.9 Scripts for boot ROM

Script:	30	Simulation time	100 us	memory Init:	0	Wave file:	0
Title	Boot ROM init - Boot ROM init						
Description	<p>The purpose of this simulation is to check initialization of the ESoCL by the embedded boot ROM</p> <p>After initialization by the boot ROM the registers are read and checked, but not written by external host simulator. Check the boot rom complete pin and the boot rom initialization status bit. After that the first untagged packet - associated with the port default VLAN ID -</p>						



from host A is broadcasted to all ports, the packet header is unmodified. The second tagged packet from host A is broadcasted to all ports, the packet header is unmodified. The third untagged packet - associated with the port default VLAN ID - from host A is broadcasted to all ports, the packet header is unmodified.

Repeat this test after temporarily modifying the `esoc_ram_2x32.mif` file in to an corrupt (=invalid address) boot rom image and repeat this test. Check the boot rom complete pin and the boot rom initialization status bit.

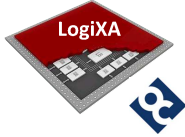
Reminder: the boot from ROM must be enabled in the `package_esoc_configuration.vhd` (constant `esoc_brom_enable`).

*(Modelsim macro > do run.do 100 0 0 30)*

### 5.5.10 Scripts for PHY management

Script:	31	Simulation time	2000 us	memory Init:	0	Wave file:	0
Title	PHY Management - SMI Management without PPU						
Description	The purpose of this simulation is to check the serial management interface of each port. Each port has a dedicated SMI interface towards an external (simulated) PHY, so no PHY Polling Unit to support multiple PHYs on one serial management interface in this simulation.						

*(Modelsim macro > do run.do 2000 0 0 31)*



### 6. ROADMAP

The roadmap of the ESoCL is shown in the table below. Please send an email to [Imaarsen@opencores.org](mailto:Imaarsen@opencores.org) if this roadmap doesn't include features that your design requires or if the expected delivery date of a feature doesn't fit in your plan.

Item	Description	Estimate of delivery date	Status
1	Support priority input and output queues, switch Ethernet packets from inbound port to outbound port through priority queues, based upon the 802.1p priority tag	Q3 - 2014	Open
2	Support CPU Ethernet Port Assignment for future items like spanning tree protocol and IGMP/MLD snooping	Q1 - 2015	Open
3	Support (rapid) spanning tree protocol		Open
4	Support VLAN Untagging/tagging of Ethernet packets that are tagged/not tagged at the inbound and/or outbound port		Open
5	Support Opencores wishbone interface		Open
6	Support Opencores Ethernet MAC		Open
7	Support Xilinx targets		Open