System Documentation
QDR Firmware

Marc Wegmueller
February 2008

Document number	1840_1005
Version	1.0
Date	20.02.2008

Client:	Paul Scherrer Institut
	5232 Villigen PSI, Schweiz
Project:	P.1840 QDR Firmware
Author:	Marc Wegmueller, Zühlke Engineering AG

Zühlke Engineering AG
Wiesenstrasse 10a
8952 Schlieren (Zurich)
Switzerland

Phone +41 44 733 6611
Fax +41 44 733 6612

info@zuehlke.com
www.zuehlke.com

Abstract

System documentation of the Quad Digital Receiver (QDR) for the Digital Beam Position Monitoring system (DBPM) at the Swiss Light Source (SLS) at Paul Scherrer Institute.

Keywords

QDR	Quad Digital Receiver
DBPM	Digital Beam Position Monitoring System
SLS	Swiss Light Source

Change History

Date	**Edition**	**Author**	**Modifications**	**Pages**
20.2.2008	1.0	mwe	Release for review	all

Distribution List

Company	**Name**	**Date**	**Signature**
PSI	P. Pollet, Dr. Th. Schilcher, Dr. B. Keil		
Zühlke	Dr. M. Wegmueller (mwe), Dr. J. Muttersbach (mut)		

Table of Contents

References

- [1] DBPM Blockdiagramm
- [2] QDR Dataflow Schema, P. Pollet, June 2000
- [3] Schemata QDRec, Rev. 1.2, P. Pollet, 10. August 2000
- [4] QDRec Firmware Requirements, P. Pollet, 29. June 2007
- [5] Modifications of QDR Module, M. Finc, 09. January 2004
- [6] Source code Main FPGA V11- Release1.113, February 2004
- [7] Source code Main FPGA V12-Release1.113, February 2004
- [8] Data sheet 8k 18 Deep Sync FIFO (CY7C4265), CYPRESS 38-06004 Rev. C 2.8.2005
- [9] Data sheet DDC (HSP50214B), Intersil FN4450.4, May 2000, revised 1.5.2007.
- [10] Data sheet VME bus (CY7C960), Cypress CY7C960, 1994, revised 4.12.1997.
- [11] VME bus Interface Handbook, Signal description VME bus, CYPRESS documentation, Sections 3 & 4.4, 1996.
- [12] Data sheet 16-Bit TRI port universal bus exchangers (SN74ABTH32316), Texas Instrument SCBS179E, 1992, revised May 1997.
- [13] IEC 821 Bus, microprocessor system bus, International Standard, 1987.
- [14] Source code Link Sharctst, P. Pollet, PSI, 2004.
- [15] Functional requirement 1840_1000, M. Wegmueller, Zuehlke Engineering AG, 2007.
- [16] Requirement specification 1840_1002, M. Wegmueller, Zuehlke Engineering AG, 2007.

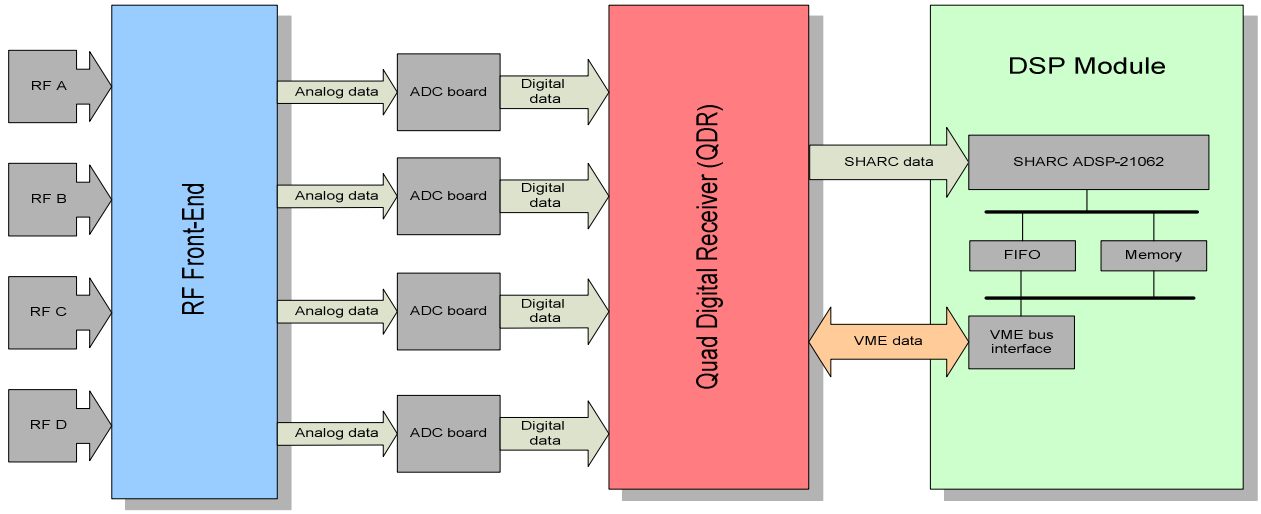
1. Introduction

1.1 Project overview

The digital beam position monitoring system (DBPM) is a core component for the safe and reliable enterprise of the Swiss Light Source (SLS) at Paul Scherrer Institute.

1.2 Situation

The Quad Digital Receiver (QDR) is an important component of the DBPM with 150 boards in use. The QDR interfaces between RF front-end and DSP module.



Basic design documentations are present and serve as basis for the firmware analysis. Two QDR versions with differences on the firmware and assembly exist. Their functional stability is insufficient (approx. a loss per month). As reasons synchronisation problems and lack of production quality are assumed.

1.3 Goal

The programming of the FPGAs on the Quad Digital Receiver (QDR ‚firmware') is analyzed and to a large extent reengineered. The goal was to arrange reliable data and control structures and thus allow the identification and distinction of hardware failure and functional deficiency. The overall system gained thereby in robustness and reliability.

Eventually, the analysis of the existing QDR firmware shall yield detailed product requirement specifications and documentation. That serves as basis for the reorganization of the firmware.

1.4 Benefits

With the review by experienced experts of Zühlke the PSI receives an impartial view on the problems of the current implementation. Documentation is provided after industry standard by regeneration of the specification. That shows the possibilities toward a future system providing more robustness, easier maintenance and portability.

1.5 Dependencies to subsystems

The novel implementation is running on the existing hardware without hardware changes. The functionality of the existing firmware served as a starting point.

For the two versions of existing hardware (v1 and v2) independent project repository are provided.

2. QDR System Overview

Nr.	Quad Digital Receiver QDR board with two FPGA	Ref.
0.1	**Control FPGA (C-FPGA)** The control FPGA configures and control the main QDR units DDC, FIFO, VME controller, monitor, and DSP-link FPGA. The C-FPGA provides the system configuration, enables dedicated data loops, allows data forwarding over the VME bus, and monitors the system status.	[15]
0.2	**Link FPGA (L-FPGA)** The link FPGA processes the four DDC data streams and interfaces with the SHARC DSP. It receives the control information from the C-FPGA.	[15]

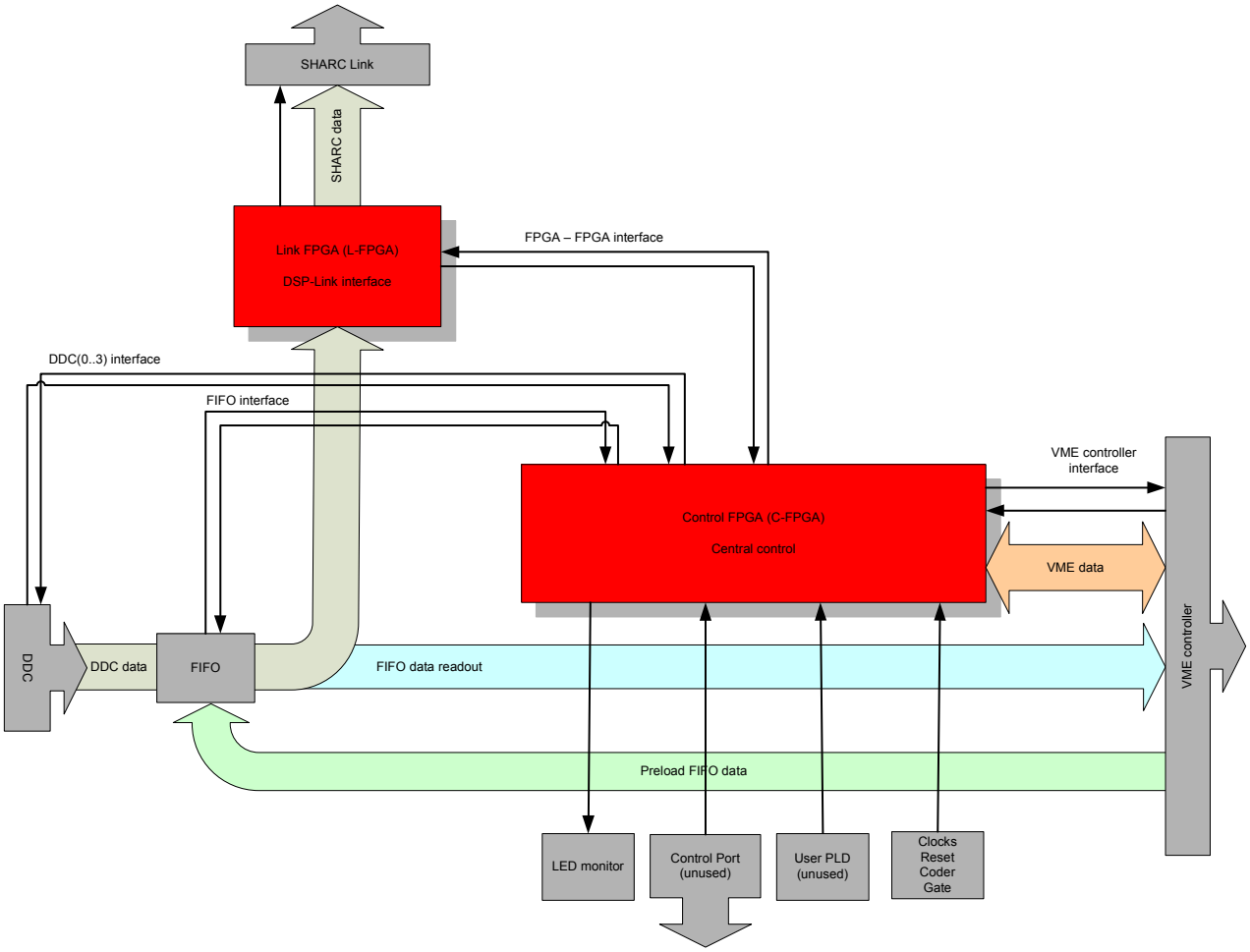


Figure 1: System overview of the QDR board with its major units: C-FPGA, L-FPGA, DDC, FIFO, VME controller, LED, and User PLD.

3. System Requirement

3.1 Functionality

Nr.	Requirements Control FPGA (C-FPGA)	Ref.
1.1	**System configuration** Configuration data is programmed via VME controller from the DSP module. The protocol is based on full-handshaking (request-acknowledgment protocol) to address and write the configuration registers located in the C-FPGA.	[1,2]
1.2	**Configuration Register** The configuration registers define the static control of DDC, FIFO, VME controller and the switches depending on the acquisition modes (DDC, VME-preloading, VME-read, and configuration mode).	[4]
1.3	**DDC parameters** Programming of the DDC configuration parameters is provided over the 16 bit processor interface bus (C_BUS).	[2,9]
1.4	**Data switches** Data can be loaded from VME into the FIFO. Their output can be routed into the L-PGA and the FIFO data can be read back over the VME bus.	[2,12]
1.5	**Monitoring of the FIFO status** The FIFO loading status is reported by dedicated flag signals to the C-FPGA.	[2,8]
1.6	**Data link control L-FPGA/SHARC** Based on the FIFO status, the DSP link FPGA (L-FPGA) is requested to forward data to the SHARC interface based on data ready/acknowledge.	[2]
1.7	**Front Panel LED** The system status is indicated through the 6 front panel LEDs.	[2]
1.8	**System programming via JTAG** The programming of the system shall be done via JTAG probing. After reprogramming the EPROM a hard reset of the system is processed.	[-]

Nr.	Requirements Link FPGA (L-FPGA)	Ref.
1.9	**Data interface with SHARC DSP (acquisition mode)** The four channels of DDC data stored the FIFO has to be processed and transmitted to SHARC DSP by transforming the four ADC data streams of total 64 bit into SHARC data protocol.	[2,4]
1.10	**L-FPGA control** The L-FPGA is triggered and controlled by C-FPGA by notification about the FIFO status. The flow control toward the DSP follows the acknowledgment protocol of the SHARK link.	[2]

3.2 Interfaces

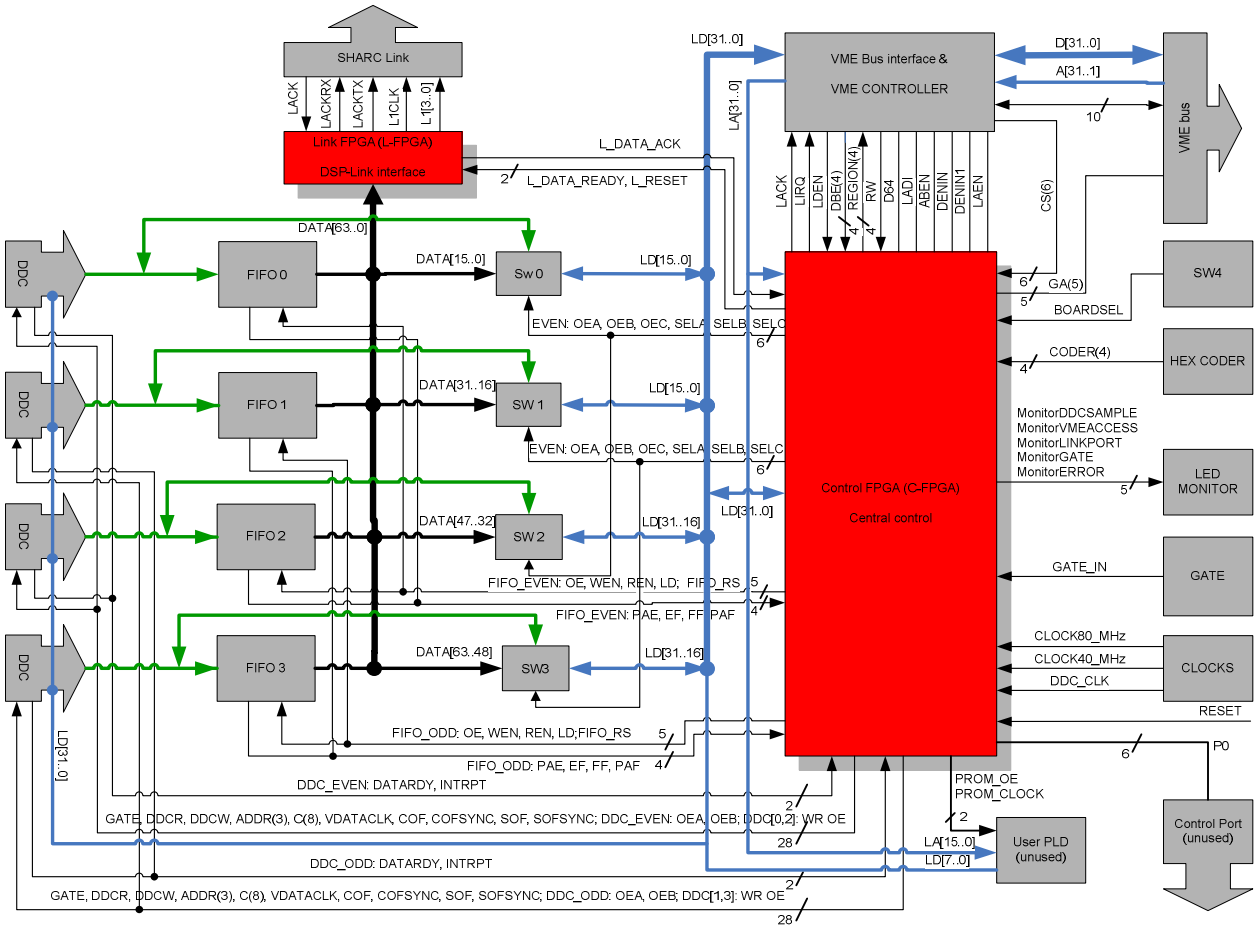


Figure 2: Interfaces and signals of C-FPGA and L-FPGA: System overview of the QDR board with its major units: C-FPGA, L-FPGA, DDC, FIFO, VME controller, LED, and User PLD.

Nr.	C-FPGA Interfaces	Ref.
2.1	**FIFO odd (FIFO 1 and FIFO 3) / even (FIFO 0 and FIFO 3) interfaces** The interface between C-FPGA and FIFO allows the exchange of status flags of the FIFO and the control of the FIFO data output toward the L-FPGA and the routing to the VME data bus. ▪ Programmable empty flag (PAE) input, odd/even ▪ Empty flag (EF) input, odd/even ▪ Full flag (FF) input, odd/even ▪ Programmable full flag (PAF) input, odd/even ▪ Output enable (OE) output, odd/even ▪ Write enable (WEN) output, odd/even ▪ Read enable (REN) output, odd/even ▪ Load programmable flag (LD) output, odd/even ▪ Asynchronous reset all (RS) output	[6,7]

Nr.	C-FPGA Interfaces	Ref.
2.2	**FIFO timing operation** The FIFO interface timing is used according to the data sheet using write cycle timing and read cycle timing specification (data sheet [8], page 7) and the flag timing specification (datasheet [8], page 9-12).	[8]
2.3	**Link FPGA interface** The signals of the L-FPGA interface control the data flow passing the L-FPGA. The protocol is based on a ready/acknowledge handshake. ■ Link data ack (L_DATA_ACK) input ■ Link data ready (L_DATA_READY) output ■ Link reset (L_RESET) output ■ FIFO status half(FIFO_HALF_FULL) (not used) ■ FIFO status empty(FIFO_EMPTY) (not used) ■ FIFO status full(FIFO_FULL) (not used)	[6,7]
2.4	**Link FPGA timing** The data is acknowledged according to a full handshake (level sensitive) with ready/ack signaling.	[-]
2.5	**VME controller** The interface with the VME controller operates the register writing and addressing. Over the VME bus data can be exchanged in both directions. ■ Data bus (LD[31..0]) bidir, 32 bit ■ Address bus (LA[31..0]) input, 32 bit ■ Read/Write (R/W) input ■ Geographical address (GA[4..0]) input, not implemented on V11, 5 bit ■ Interrupt request (LIRQ) output ■ Delay response (LACK) output ■ Data byte enable (DBE[3..0]) input, 4 bit ■ Chip select (CS[5..0]) (not used) ■ Region select(REGION[3..0]) output, 4 bit ■ Latch address enable (LAEN) (not used) ■ Address bus enable (ABEN) (not used) ■ Latch data enable (LADI) (not used) ■ (DENIN) (not used) ■ (DENIN1) (not used) ■ Data access flag (D64) (not used) ■ TP7 connector (LDEN) input	[2, 6, 7] [10,11,13]

Nr.	C-FPGA Interfaces	Ref.
2.6	**VME timing** The VME bus is used in accordance with the VME bus standard. The VME bus is operated in I/O mode (chapter 3.9.9, page 3-85) using ■ Single Cycle Read access (chapter 3.12, page 3-129), ■ Single Cycle Write access (chapter 3.12, page 3-130), and ■ Single Cycle Read-Modify-Write access (chapter 3.12, page 3-133).	[11]
2.7	**DDC(0..3) odd/even** The interface to the DDC configures and controls the 4 DDC units located on each QDR board. Their parameters are loaded into the DDCs over the processor interface bus (C_BUS). The COF/SOF control signals shall allow the tuning of the offset frequency compensation. ■ DDC clock for all DDC (DDC_CLK) (not used) ■ DDC output enable (OEA, OEB) output, odd/even ■ DDC data ready (DATARDY) input, odd/even ■ DDC interrupt (INTRPT) input, odd/even ■ DDC(1..4) bus data direction (W/R) output, 4 bit ■ DDC (1..4) bus driver output (OE) output, 4 bit ■ DDC bus read latch (DDCR) output ■ DDC bus write latch (DDCW) output ■ DDC bus addresses (ADDR[2..0]) output ■ DDC bus data (C[7..0]) output,8 bit ■ DDC (VDATACLK) output ■ Select for SOF & COF (SOFCOFSEL[1..0]) (currently not used) ■ DDC COF sync clock (COFSYNC) (currently not used) ■ DDC COF parameter (COF) (currently not used) ■ DDC SOF sync clock (SOFSYNC) (currently not used) ■ DDC SOF parameter (SOF) (currently not used) ■ Gate output (GATE) output, gated or interpolated	[6,7]
2.8	**DDC timing** The DDC unit is operated in the 8-bit uProcessor interface mode (data sheet, pages 40ff).	[9]
2.9	**Switches odd/even** The switches enable the three data transfer modes: SHARC data, preload FIFO data, and FIFO data readout. ■ Driver output enable (OEA, OEB, OEC) output, odd/even ■ Driver select (SELA, SELB, SELC) output	[6,7, 12]

Nr.	C-FPGA Interfaces	Ref.
2.10	**Clocks, reset** The two system clocks are provided at 80 MHz and 40 MHz. ■ Clock (CLOCK80) input, 80MHz ■ Clock (CLOCK40) input, 40MHz, dual clock domain, ■ Reset (RESET) input	[6,7]
2.11	**Coder** The board address is defined in version 12 by a rotational switch. This coder provides a 4 bit address identifying the board. In version 11 the coder is not used. ■ Coder (CODER[3..0]) input, not implemented on V11, 4 bit ■ Board select (BOARDSEL) input, not implemented on V12	[6,7]
2.12	**Gate** The gate represents the external trigger for synchronization of the data acquisition. ■ External gate input (GATEIN) input	[6,7]
2.13	**Monitor** The C-FPGA provides status information by 5 LED on the front panel reporting failure of the 5 major system units (DDC, VME, DSP-Link, Gate, and entire system). ■ DDC data ready monitor (DDCSAMPLE) output, green LED, L2, DDC data ready ■ VME access monitor (VME ACCESS) output, data ack of VME ■ Link port access monitor (LINKPORT) output, DSP data ack ■ Gate acquisition monitor (GATE) output, 2 LED states, red inactive, green active ■ Overall board error (ERROR) output	[6,7]
2.14	**Control Port** Port P0 was intended to be used for configuration but was not implemented. Version V12 multiplexes the same signal ports for its address configuration (board addressing 5.2). ■ P0 (not implemented)	[6,7]
2.15	**User PLD** The user PLD would allow a user defined configuration stored locally on the QDR board. ■ Data bus (LD[7..0]) bidir, 8 bit ■ Address bus (LA[15..0]) input, 16 bit ■ PROM Output enable (PROM_OE) (not used) ■ PROM clock (PROMCLOCK) (not used)	[6,7,15]
2.16	**Debug signal** The soft reset provides a kill signal that is externally fed back to the reset unit shutting down the system. ■ Soft reset (SOFTRESET) output	[6,7]

Nr.	L-FPGA Interfaces	Ref.
2.17	**Control FPGA** The L-FPGA is controlled over the interface with the C-FPGA. The data flow between DDC and DSP is managed by a ready/acknowledge protocol. ■ Link data ack (L_DATA_ACK) output ■ Link data ready (L_DATA_READY) input ■ Link reset (L_RESET) input ■ FIFO status half(FIFO_HALF_FULL) (not used) ■ FIFO status empty(FIFO_EMPTY) (not used) ■ FIFO status full(FIFO_FULL) (not used)	[6,7]
2.18	**SHARC** The 64 data bits received from the 4 DDC units are processed into the SHARC DSP data format. ■ LACK input ■ LACKRX output ■ LACHTX output ■ L1CLK output ■ L1[3..0] output	[14]
2.19	**FIFO** The data of the 4 FIFO units is available parallel as 64 data bits. ■ Data (DATA) input, 64 bit	[8]

3.3 Operation modes and system configurations

Nr.	Mode	Ref.
3.1	**Data Transmission Modes** The system is used in 3 dedicated data transmission modes: A DDC data is transferred through the FIFOs to the SHARC processor. B The FIFOs are preloaded with data from the VME bus. C The FIFO data is read-out and transferred over the VME bus.	[4]
3.3	**Reset** ■ The FIFO can be reset with writing on address 0x000C (FIFO_RS) ■ The FPGA and the VME interface are reset on flag [15] @ address 0x0010. The FIFO values are unchanged. ■ Init modes are all modes 0 (VME access mode disabled, disabled acquisition, FIFO forwarding mode 0, disabled link port)	[-]

Nr.	Mode	Ref.
3.10	**FIFO Status Flags (address 0x0020, FIFO status interface)** DDC parameters shall be configured as static control. 0 ODD FIFO Empty Flag (EF) 1 ODD FIFO Programmable Empty Flag (PAE) 2 ODD FIFO Programmable Full Flag (PAF) 3 Odd FIFO Full Flag (FF) 4 Even FIFO Empty Flag (EF) 5 Even FIFO Programmable Empty Flag (PAE) 6 Even FIFO Programmable Full Flag (PAF) 7 Even FIFO Full Flag (FF) 8 Gate Monitor 9 FIFO busy flag 10 Link data acknowledge	[2]
3.12	**Global Address and Version Number (address 0x0080)** [31..24] "Q" 23 "0" [22.18] Slot number (Global Address) [17..16] "0" [15..0] Version number	[2]
3.13	**Interrupt Registers (address 0x0100, IRQ status flags): Prioritization as IRQ vector** ■ IRQ4 status flag: lowest priority ■ IRQ3 status flag ■ IRQ2 status flag ■ IRQ1 status flag: highest priority	[2]
3.14	**FIFO Flags** ■ X register: IRQ flags for FIFO status register ■ M register: Mask flags ■ V register: Programmable IRQ vector	[2]

3.4 Board addressing: Slot geographical address map

Nr.	Board addressing	Ref.
4.1	**Fixed geographical address in V11** The board address in version 11 is hard-coded in the FPGA (user specification)	[6]
4.2	**Geographical address by switch in V12** A 16 position rotational switch gives the address range by 5 bits in version 12.	[7]

3.5 Timing constraints

Nr.	Design Constraints	Ref.
5.1	**System Timing** ■ 80 MHz: System clock, interface clock for VME operations (clock skew 1+1.2 ns) ■ 40 MHz: Interface clock to DDC, FIFO (regular + 2.5ns delayed), and SHARC (clock skew 0.5ns) ■ Clocks are derived from a common 40 MHz on-board oscillator.	[4]
5.2	**DDC Timing** ■ DDC operate in a dual clock domain (27 MHz and 40 MHz) while towards the two FPGA all data streams operate at 40 MHz. ■ Variable FIFO data sampling with 4 kHz up to 1 MHz.	[-]
5.3	**System Latency** The reading out of a complete FIFO content must be processed within 330ms up to the DDC sampling rate of 1 MHz.	[-]

3.6 System additions

Nr.	Features not yet implemented in the current version	Ref.
6.3	SOF / COF functionality: Writing carrier offset frequency and re-sampler offset frequency via the C-FPGA (COF, COFSYNC, SOF, and SOFSYNC).	[9]
6.4	DDC data format with full functionality of DDC data format providing magnitude, phase and both magnitude&phase (SEL[2..0]).	[9]
6.5	Interrupt operation with priority on IRQ 1 to 4: IRQ triggered on VME bus are processed (depending on the IRQ registers and the IRQ signal lines LDEN and LIRQ) to show the masked FIFO status.	[5,6,7]

3.7 Overview of the board timing environment

The on board timing are not relaxed. Therefore, the FPGA design are implemented with care on the interface timing. When ever possible registers close to hardware interfaces are incorporated and the synthesis timing is slightly overcontraint.

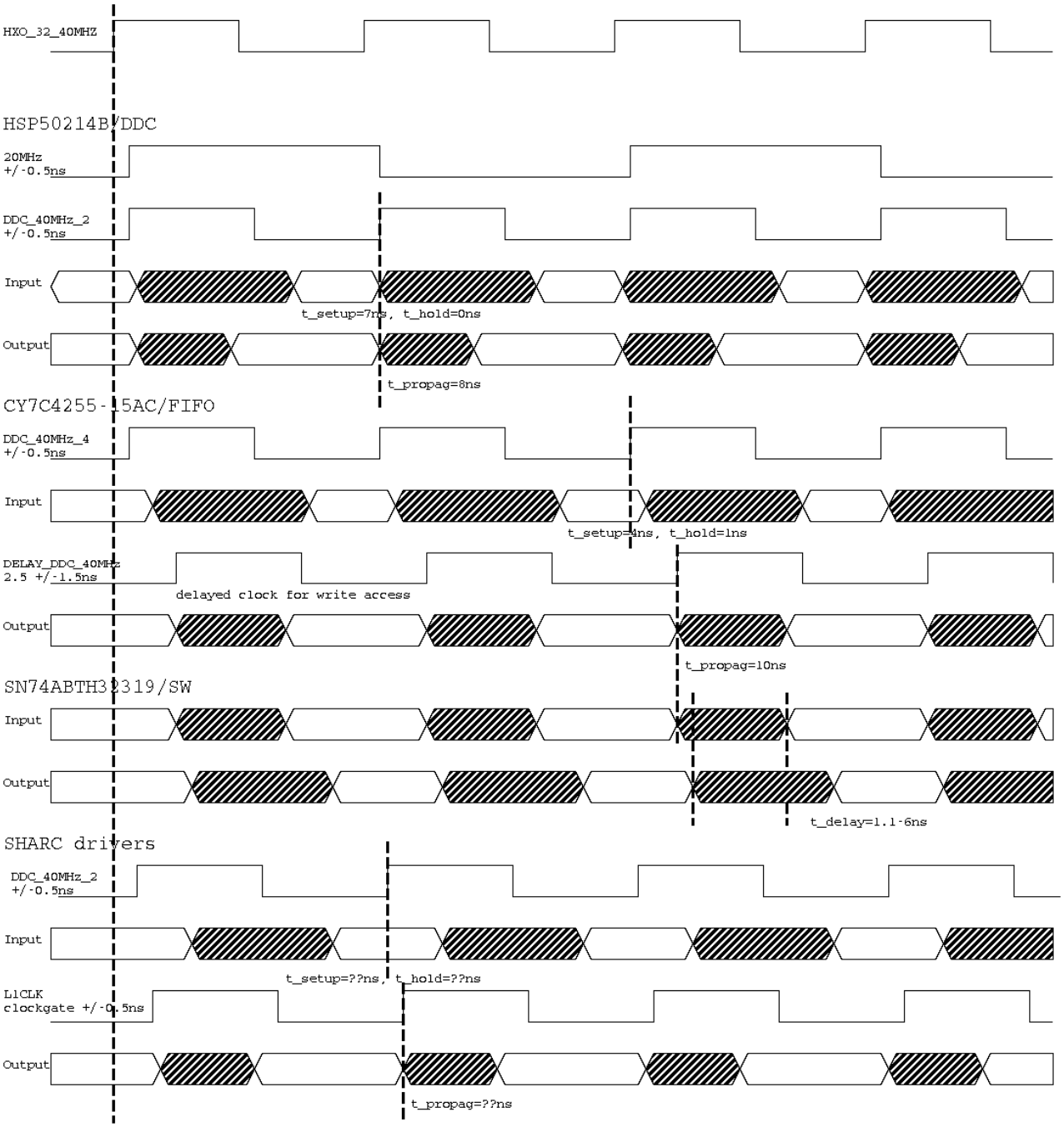


Figure 3: Current timing environment of the QDR clocks and interfaces.

4. Design Constraints

Nr.	HW / SW utilization	Ref.
7.1	**Hardware** ▪ QDR board: Version 11 and Version 12 ▪ Control C-FPGA: Altera EPF10K30 ▪ Link L-FPGA: Altera EPF10K10	[3]
7.2	**Software tools** ▪ Altera design flow: QARTUS II Version 7.2 Web Edition ▪ Modelsim Altera Web Edition 6.1g	[-]

5. Control FPGA (C-FPGA)

The C-FPGA consists of the seven major units shown in Figure 4.

- The main controller *Central Control* contains the control status of the system *ControlStatusxS* and hands over the control signals. The various interfaces to the dedicated hardware are modular enclosed in each their own unit, e.g. *SHARC, VME*, general Interfaces (*DDC-FIFO-Selectors*), *GateSynchronization, BoardAddress*, and *Monitor LED*.
- The *VME Interface* unit stores all the IRQ register and the configuration register RegisterxD provided by the remote control unit through the VME interface bus.
- The *Gate Synchronzation* unit provides the system with the appropriate trigger signal depending on the system configuration. Given the acquisition mode, the other interfaces are enabled and data storage controlled.
- The *Board Address* unit detects whether the board is selected and whether data has to be provided over the VME bus. Furthermore the appropriate bus response is generated.
- The *SHARC interface* unit processes the hand-shake signals to the L-FPGA and interfaces with the remote state machine to process the data delivering of the SHARC link path.
- The *DDC-FIFO-Selector* unit provides the external interface devices with control signals.
- The *Monitor* unit simply collects the corresponding information to control the monitor LED

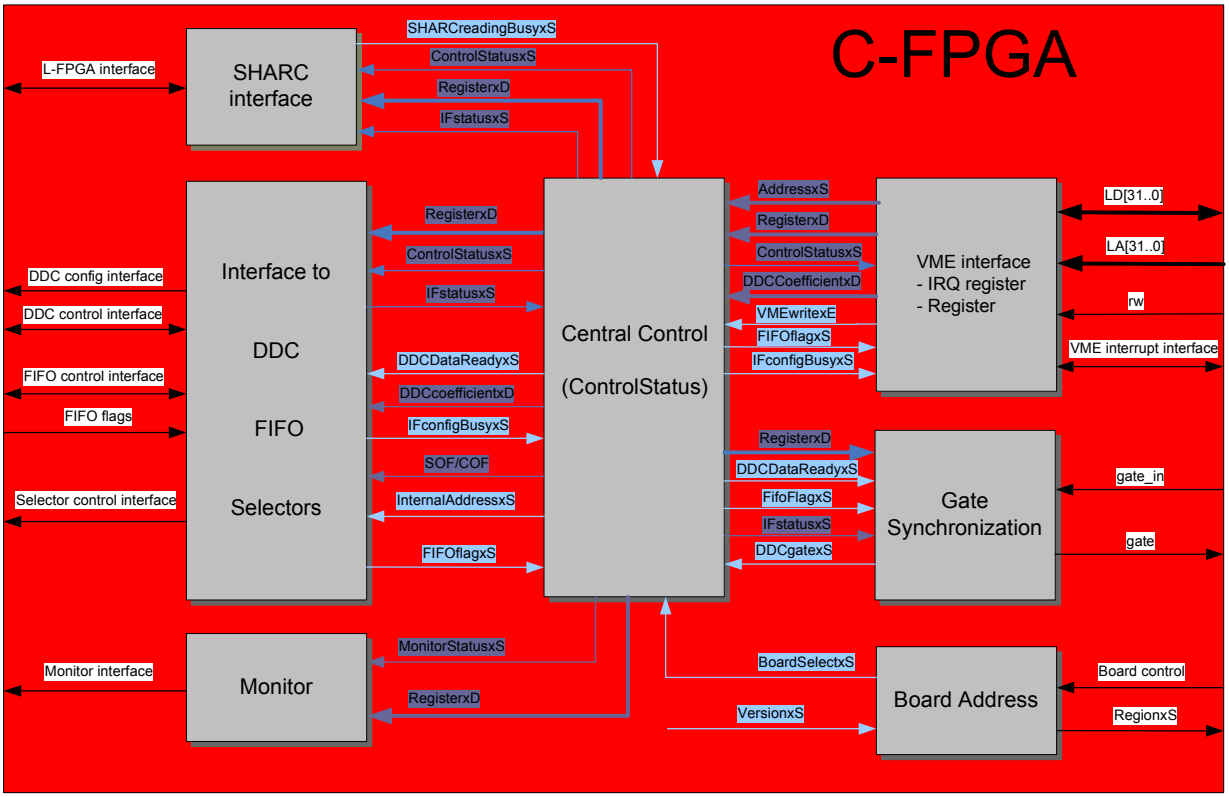


Figure 4: Main unit of C-FPGA: Central control operates as master of the remaining interface units.

The signals *ControlStatusxS*, *RegisterxD*, and the collection of status signals *IFstatusxS* are distributed through out the entire C-FPGA by the *Central Control* unit giving the overall state of the system. Using these status informations the depending units calculate their own states, output signals, and a additional control signal. These signals fed back to the *Central Control* unit for further distribution, e.g. *DDCconfigBusyxS, DDCDataReadyxS, FIFOflagxS, AddressxS, DDCcoefficientxD.*

5.1 Central Control Unit

Over the Address of the VME the system state is defined and stored in the *ControlStatusxS* register. Its values are defined in the package const.vhd. The *ControlStatusxS* serves as the base of all the other slave units and gives the global status.

Central Control Unit interface		
Inputs:	RegisterxS[15..0]	Command Register stored in the Register Map
	AddressxS[31..0]	Address selected by VME
	BoardSelectxS	Board selection
	VMEwritexE	VME write enable
	SHARCreadingBusyxS	Blocking while SHARC processing
	FIFOflagxS[10..0]	FIFO flags
	IFstatusxS	Interface status
Outputs:	ControlStatusxS	System state defined in point 6.1
	InternalAddressxS[7..0]	Address generation for DDC C_BUS
	DDCcoefficientxD[31..0]	DDC coefficients
	SystemDataxD	
	SoftResetxRBO	Soft reset

The contral state is evaluated depending on the *AddressxSI* as given in the following table.

Control States of the central unit (ControlStatusxS)

- **Default state:** *idle*
  Nothing is processed, the system remains idle with default outputs.
- *ddcFIFOSHARCwrite*
  Data transfer from DDC over FIFO to SHARC interface (Data transmission mode A).
- *VME_oddFIFOwrite, oddFIFO_VMEread,* address 0x0000
  VME_evenFIFOwrite, evenFIFO_VMEread, address 0x0004
  VMEbothFIFOwrite address 0x0008
  Write data from VME to FIFO or read in the opposite direction (data transmission mode B while writing and data transmission mode C while reading).
- *bothFIFOreset*
  FIFO reset, address 0x000C
- *writeCommandRegister, readCommandRegister*
  Write and read command register (address0x0010)
- *readFIFOflags*
  Read FIFO flags (address 0x0020)
- *writeStaticDataRegister*
  Write static DDC register (address 0x0040). Blocking until *DDCconfigBusyxS* released.
- *readBoardStatusAndVersionNr*
  Read board status and version number (address 0x0080)
- *readX1register, readX2register, readX3register, readX4register*

Control States of the central unit (ControlStatusxS)
Read interrupt register X1, X2, X3, X4 (address 0x0104, 0x0114, 0x0124, 0x0134)
▪ *readM1register, readM2register, readM3register, readM4register* Read interrupt mask register M1, M2, M3, M4 (0x0108, 0x0118, 0x0128, 0x0138)
▪ *readV1register, readV2register, readV3register, readV4register* Read interrupt register V1, V2, V3, V4 (address 0x010C, 0x011C, 0x012C, 0x013C)
▪ *readIEregister* Read IE register (address 0x0140)
▪ *writeFNregister, readFNregister* Write/Read IRQ status register (address 0x0100)
▪ *writeX1register, writeX2register, writeX3register, writeX4register* Write interrupt register X1, X2, X3, X4 (address 0x0104, 0x0114, 0x0124, 0x0134)
▪ *writeM1register, writeM2register, writeM3register, writeM4register* Write interrupt mask register M1, M2, M3, M4 (0x0108, 0x0118, 0x0128, 0x0138)
▪ *writeV1register, writeV2register, writeV3register, writeV4register* Write interrupt register V1, V2, V3, V4 (address 0x010C, 0x011C, 0x012C, 0x013C)
▪ *writeIEregister* Write IE register
▪ *writeSOF1, writeSOF2, writeSOF3, writeSOF4, writeSOFall* Write SOF DDC register (address 0x0200, 0x0204, 0x0208, 0x020C, 0x020F). Blocking until *DDCconfigBusyxS* released.
▪ *writeCOF1, writeCOF2, writeCOF3, writeCOF4, writeCOFall* Write COF DDC register (address 0x0400, 0x0404, 0x0408, 0x040C, 0x040F). Blocking until *DDCconfigBusyxS* released.
▪ *readPAEODDregister, readPAFODDregister* Read alternating PAE / PAF register of the odd FIFO (address 0x0800, 0x804).
▪ *readPAEEVENregister, readPAFEVENregister* Read alternating PAE / PAF register of the even FIFO (address 0x0808, 0x80C).
▪ *writePAEODDregister, writePAFODDregister* Write alternating PAE / PAF register of the odd FIFO (address 0x0800, 0x0804).
▪ *writePAEEVENregister, writePAFEVNEregister* Write alternating PAE / PAF register of the even FIFO (address 0x0808, 0x080C).
▪ *writeDDC1parameter, writeDDC2parameter, writeDDC3parameter, writeDDC4parameter* Write DDC parameters (address 0x1000, 0x2000, 0x4000, 0x8000) over the configuration bus (C_BUS).
▪ *writeAllDDCparameter* Write all DDC parameter (address 0xF000) over the configuration bus (C_BUS).

As special cases depending on input signals are handled

- The signal *DDCconfigBusyxSI* blocks the control state in its actual state.
- The default state while inactive *BoardselctxSI* is *idle.*
- Read and write separation is dependent on the *VMEwritexSI.*

5.2 Board Address Unit

Encoder, global address, and predefined defined signals are compared with actual VME address to detect valid board addressing and set appropriate the region signal on the board interface and the internal board select signal (Figure 5).

Board Address Unit interface		
Inputs:	LAxDI[31..0]	VME address defining the actual status according to 4.1
	GAxDI[4..0]	Global address as part of the VME bus interface
	P0XSI[4..0]	Remapped GAxD on V11
	DBExDI[3..0]	Date byte enable (default all '1')
	CSxSI	Chip select specifying the I/O mode
	CoderxSI[3..0]	Rotational encoder data for range selection on LAxD
Outputs:	RegionxSO[3..0]	VME acknowledge of activation
	BoardSelectxSO	System flag for VME board activation

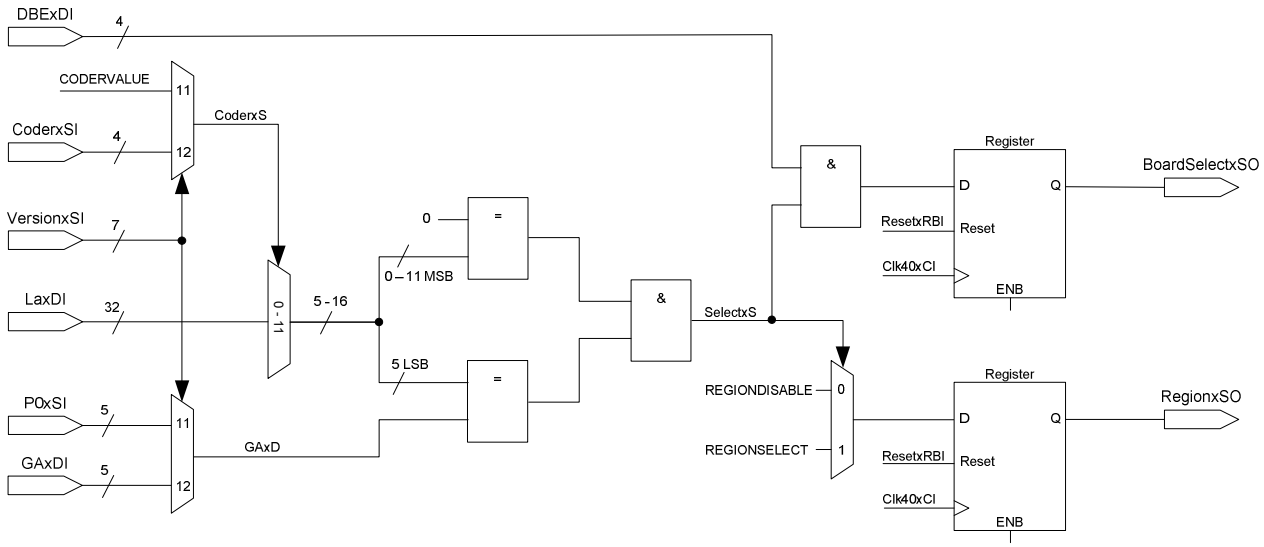


Figure 5: Schematic of *Board Address* unit generating the *BoardSelectxSO* signal for internal use and *RegionxSO* as external bus response depending on the input signals *CoderxSI*, *P0xSI*, *GAxDI,* and *DBExDI* compared with the VME bus data *LAxDI*.

The differences between the two board version are identified with Version 11 and Version 12.They distinguish on the hardware board assembly.

Board version differences	
Version 11:	▪ Input of the configuration port *P0xSI* is remapped to the internal global address *GAxD*. ▪ Input *CoderxSI* is unused, the value for internal *CoderxS* remains constant with CODERVALUE.
Version 12:	▪ Input GAxSI gives the internal global address *GAxD*. ▪ Input *CoderxSI* gives the coder value *CoderxS*.

QDR Implementation Version is labelled V2.0. This can be identified in the *board status and version number* information.

5.3 VME Interface Unit

The asynchronous data exchange with address *LAxD* and data *LDxD* bus is controlled according to the VME specification using enable and acknowledge as well as the interrupt request.

The data is written and read over the *LDxDI* and *LDxDO* bus. Out of the register map dedicated signals are derived and provided to the remaining units of the system.

VME Interface Unit interface		
Inputs:	LAxDI[31..0]	VME Address defining the access according to 4.1
	LDxDI[31..0]	Data bus input
	RWxSI	Read or write operation
	LDENxSBI	Local data enable
	ControlStatusxS	System status
	IFconfigBusyxS	DDC and FIFO blocking Lack during parameter configuration
	FifoFlagxS[10..0]	FIFO flags
Outputs:	LIRQxSBO	Local interrupt request
	LackxSBO	Bus acknowledgment of a local transfer cycle
	LDxDO[31..0]	Data bus input
	VMEwritexE	VME write enable
	AddressxS[15..0]	Address selection decoded out of LAxD
	RegisterxS[15..0]	Command register
	DDCcoefficientxD[31..0]	DDC coefficients

Figure 6 shows the schematic of the VME architecture. Given the dedicated *ControlStatusxSI* the VME data *LDxDI* is stored in the corresponding register. So the interrupt register X, M, V, and IE are stored and read back. Using as addtion *StatusAndVersionNrxS* and *GAxD*; *VersionxSI* and *IRQstatusxS* the output data can be calculated and provided as *LDxDO*. Not intermediate stored but selected out of the VME data are *SOFxD*, *COFxD* and the *DDCcoefficientxD*.

The VME link interrupt *LIRQxSB* depends on the two interrupt contribution *LIRQoutxS* and *LIRQoffxSP*:

- *LIRQoutxS* is set by an IRQ status unequal to zero. Such an IRQ is initialized by the FIFO flags, masked with the M and X flags, and enabled with the interrupt enable flag IE. The reset of such an IRQ is done with a VME bus write to the corresponding IRQ flag; depending on the control state *ControlStatusxSI* and the value *VMEdataInxD(3..0).*
- *LIRQoffxSP* depends on the *LDEN* and *LDxDI* with a timeout option.

The IRQ status can be monitored as the corresponding IRQ vector *VxD* observed as data vector *IRQvectorxD* read via the VME bus.

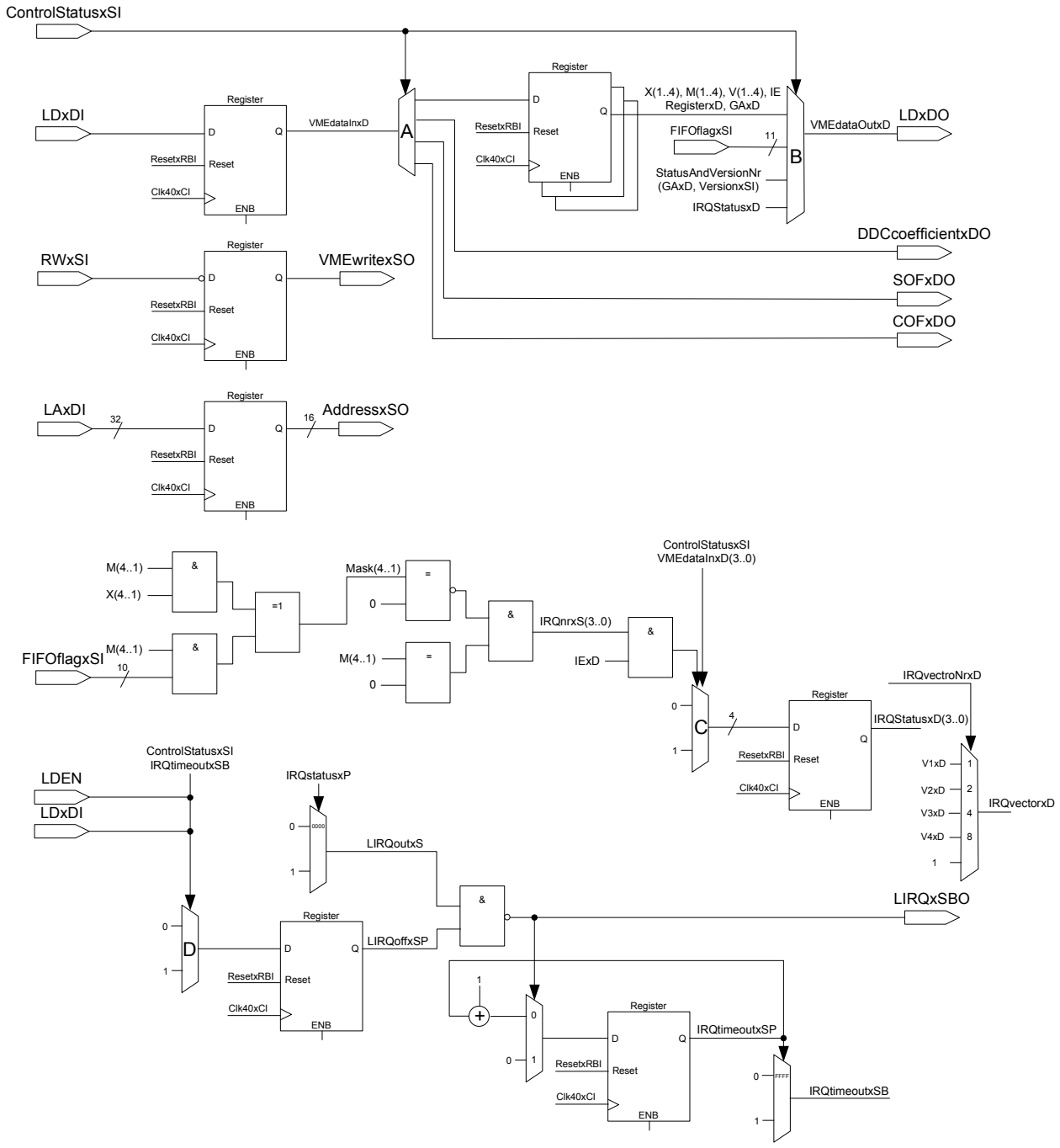


Figure 6: Schematic of *VME Interface* unit: Over the VME bus the register map can be written and read using the register *X, M, V IE RegisterxD* and *GAxD* as well as data forwarding of data to other system units, e.g. *SOFxD*, *COFxD*, *DDCcoefficientxD*; furthermore the VME unit handles the IRQ processing dependent on the masked FIFO flags *FIFOflagxSI*, VME signals *LDEN* and *LDxDI*, timeout, and the system status *ControlStatusxS*.

The functionality of the selectors A,B, C, and D in Figure 6 are given in the following tables:

Selector A in Figure 6	
ControlStatusxSI	**store or forward VMEdataInxD in**
writeCommandRegister	RegisterxS
writeBoardStatusAndVersionNr	GAxD
writeX1register	X1xD
writeX2register	X2xD
writeX3register	X3xD
writeX4register	X4xD
writeM1register	M1xD
writeM2register	M2xD
writeM3register	M3xD
writeM4register	M4xD
writeV1register	V1xD
writeV2register	V2xD
writeV3register	V3xD
writeV4register	V4xD
writeIEregister	IExD
writeCOF1	COFxDO
writeCOF2	COFxDO
writeCOF3	COFxDO
writeCOF4	COFxDO
writeCOFall	COFxDO
writeSOF1	SOFxDO
writeSOF2	SOFxDO
writeSOF3	SOFxDO
writeSOF4	SOFxDO
writeSOFall	SOFxDO
writeDDC1parameter	DDCcoefficientxDO
writeDDC2parameter	DDCcoefficientxDO
writeDDC3parameter	DDCcoefficientxDO
writeDDC4parameter	DDCcoefficientxDO
writeAllDDCparameter	DDCcoefficientxDO

Selector B in Figure 6	
ControlStatusxSI	**VMEdataOutxD**
Default:	IRQvectorexD
readCommandRegister	RegisterxS
readFIFOflags	FIFOflagxSI
readBoardStatusAndVersionNr	X"51", not(GAxD)
readFNregister	IRQstatusxD
readX1register	X1xD
readX2register	X2xD
readX3register	X3xD
readX4register	X4xD
readM1register	M1xD
readM2register	M2xD
readM3register	M3xD
readM4register	M4xD
readV1register	V1xD
readV2register	V2xD
readV3register	V3xD
readV4register	V4xD
readIEregister	IExD

Selector C in Figure 6				
ControlStatusxSI	**VMEdataInxD**	**IRQnrxS**	**IExD**	**IRQstatusxD**
Default:	-	-	-	IRQstatusxD
writeFNregister	'1'	-	-	0
-	-	'1'	'1'	'1'

Selector D in Figure 6				
ControlStatusxSI	**LDENxSB**	**IRQtimeoutxS**	**not(LDxDI = 0)**	**LIRQoffxSP**
Default:	-	-	-	LIRQoffxSP
-	'0'	'0'	-	'0'
writeFNregister	-	-	'1'	'1'

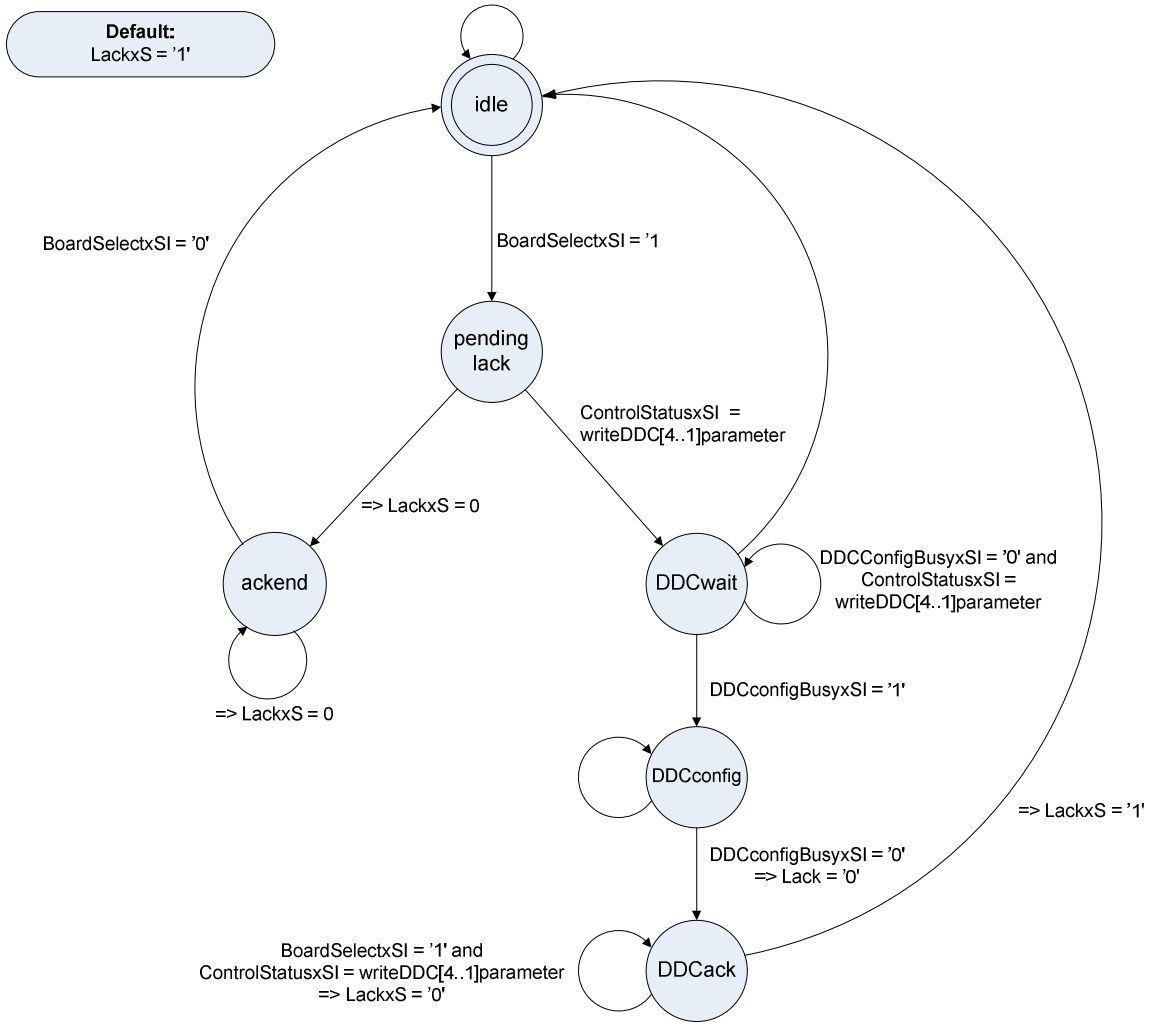


Figure 7: State flow diagram of VME status for generation of the VME bus acknowledge *LackxS* (special treatment of DDC configuration to delay the *LackxS* returns to 1 until the configuration is finished independent on the VME bus traffic).

The state machine of the VME unit is shown in Figure 7. *LackxS* is high by default, acknowledging every cycle, that basically indicates an idle VME interface. After reset the state machine is initialized to the *idle* state (labeled with the double circle). A board selection indicated with the signal *BoardSelectxSI* triggers the state machine. Depending on the control status *ControlStatusxSI* the acknowledgment is stopped after the state *pendinglack* or after writing the DDC parameter in state *DDCwait* and *DDCconfig* (having first waited for a *DDCconfigBusyxSI*).

VME Interface unit together with the *Board Address* unit build the only interface to the VME backplane. Therefore, the asynchronous bus has been completely isolated from the remaining system; the main controller *CentralControl* and the complete data forwarding is handled by the VME interface. Input and output registers are relaxing the board timing, The asynchronous bus is converged to the synchronous island of the QDR system. The obtained latency of 2 full clock cycles in the longest case is no problem due to the asynchronous VME bus specification without clock.

5.4 Interface Unit to DDC, FIFO, and Selectors

Interface Unit DDC interfaces			
Input	ControlStatusxS	System status	
	RegisterxD[15..0]	Command register for extracting monitor functionality	
	DDCcoefficientxD[31..0]	DDC coefficient derived from the register map	
	DDCDataReadyxSBI	New data ready at the DDC output	
	InternalAddressxS[7..0]	Internal address to be applied on the C_BUS	
Output	DDCoutAxEBO[1..0]	DDC port A enable	odd and even FIFOs
	DDCoutBxEBO[1..0]	DDC port B enable	odd and even FIFOs
	DDCconfigxEBO[3..0]	Configuration bus driver enable	every FIFO
	DDCconfigWRxSBO[3..0]	Configuration bus direction control	every FIFO
	DDCwxSBO	Processor interface write strobe	
	DDCrxSBO	Processor interface read strobe	
	VDataClkxCO	Gated clock for external DFF	
	AddrxSO[2..0]	Processor interface address bus	
	CxDO[7..0]	Processor interface data bus	
	DDCsofcofxSO[1..0]	DDC selector for SOF/COF configuration	
	DDCcofsyncxSO	Carrier offset frequency sync	
	DDCcofxDO	Carrier offset frequency	
	DDCsofsyncxSO	Re-sampler offset frequency sync	
	DDCsofxDO	Re-sampler offset frequency	
	IFconfigBusyxSO	Stall central control while writing the DDC, SOF/COF, or PAE/PAF configuration	

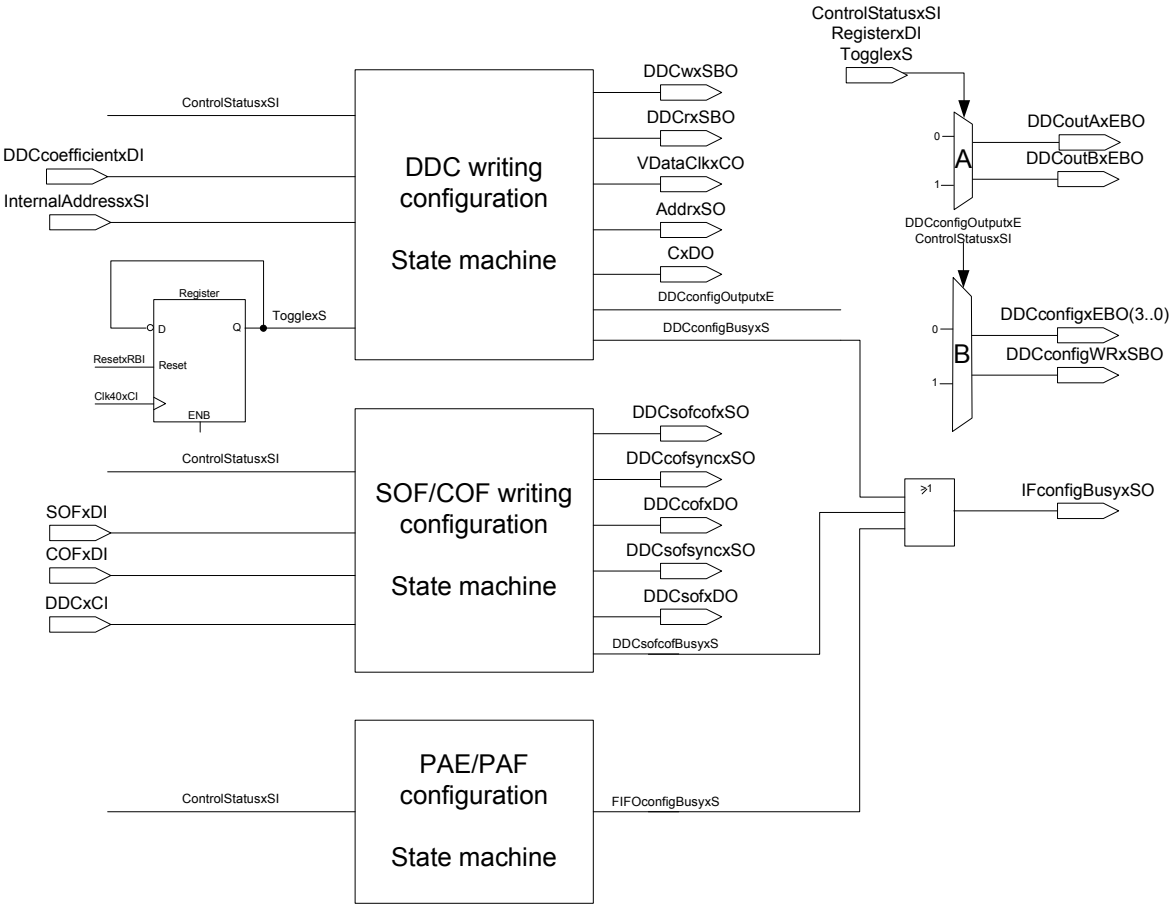


Figure 8: Schematic of *DDC Interface* unit with the basic selectors for the regular DDC interface and the two state machine units for *DDC writing configuration* and *SOF/COF writing configuration*.

As shown in Figure 8 the DDC interface contains selectors of two DDC data ports, DDC configuration interface (C-BUS), and SOF/COF configuration data. The selectors for the enable signal generation (*DDCoutAxEB* and *DDCoutBxEB*) and the configuration enable signal (*DDCconfigxEB* and *DDCconfigWRxSB*) depend on *ControlStatusxSI* and *DDCconfigOutputxE*.

Selector A in Figure 8			
ControlStatusxSI	**RegisterxDI(9..8)**	**DDCoutAxEB**	**DDCoutBxEB**
Default:		'1'	'1'
DDC_FIFO_SHARCwrite	'00'	'0'	'1'
DDC_FIFO_SHARCwrite	'01'	'1'	'0'
DDC_FIFO_SHARCwrite	-	TogglexS	not(TogglexS)
idle	'00'	'0'	'1'
idle	'01'	'1'	'0'
idle	--	TogglexS	not(TogglexS)

Selector B in Figure 8				
DDCconfigOutputxE	**ControlStatusxSI**	**index**	**DDCconfigxEB(index)**	**DDCconfigWRxSB(index)**
Default:	-	1,2,3,4	'1'	'1'
'1'	writeDDC1parameter	1	'0'	'1'
'1'	writeDDC2parameter	2	'0'	'1'
'1'	writeDDC3parameter	3	'0'	'1'
'1'	writeDDC4parameter	4	'0'	'1'
'1'	writeAllDDCparameter	1,2,3,4	'0'	'1'

The two interfaces C-Bus, for DDC writing configuration, and SOF/COF, for writing SOF/COF configuration, are implemented both as state machines.

- The C-bus signals (*DDCwxSB, DDCrxSB, VDataClkxC, AddrxS* and *CxD* shown in state machine of Figure 9) depend on the *ControlStatexS, DDCcoeffientxD* and *InternalAddressxS*. Its state is only updated with half the system clock.
- The SOF/COF state machine combines parallel inputs to serial outputted SOF/COF configuration bits (Figure 10).

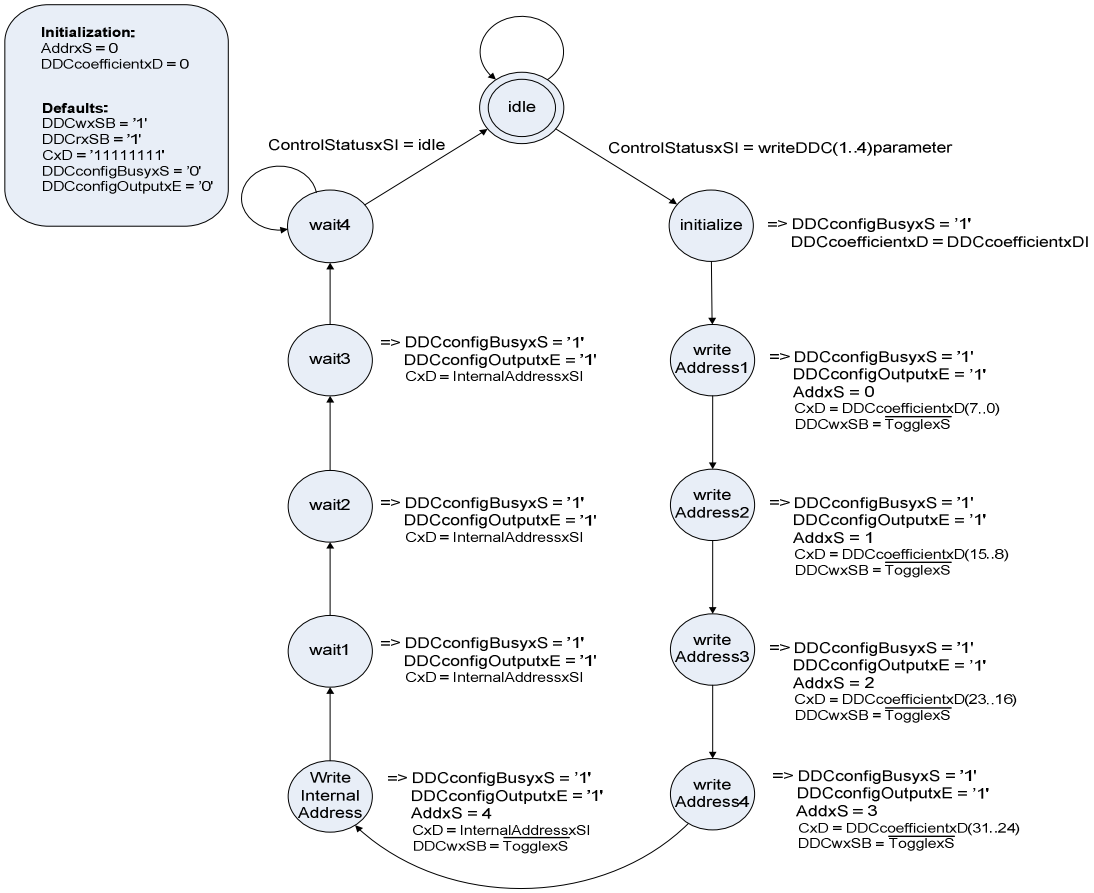


Figure 9: State flow diagram of DDC configuration with initialization values after reset, default values and the consecutive DDC configuration sequence *Address1* till *Address4*, *InternalAddress*, and three *wait* cycles. The state machine operates on half the system clock using a toggle flip-flop as state update enable.

As shown in Figure 10 depending on the *ControlStatexSI* SOF or COF information is stored and the SOF/COF signal demultiplexer unit is configured. Either SOF or the COF information is serial transmitted up to a length of 32 bits. *DDCsofcofBusyxS* is used for blocking the *Central Control* unit during this critical configuration procedure.

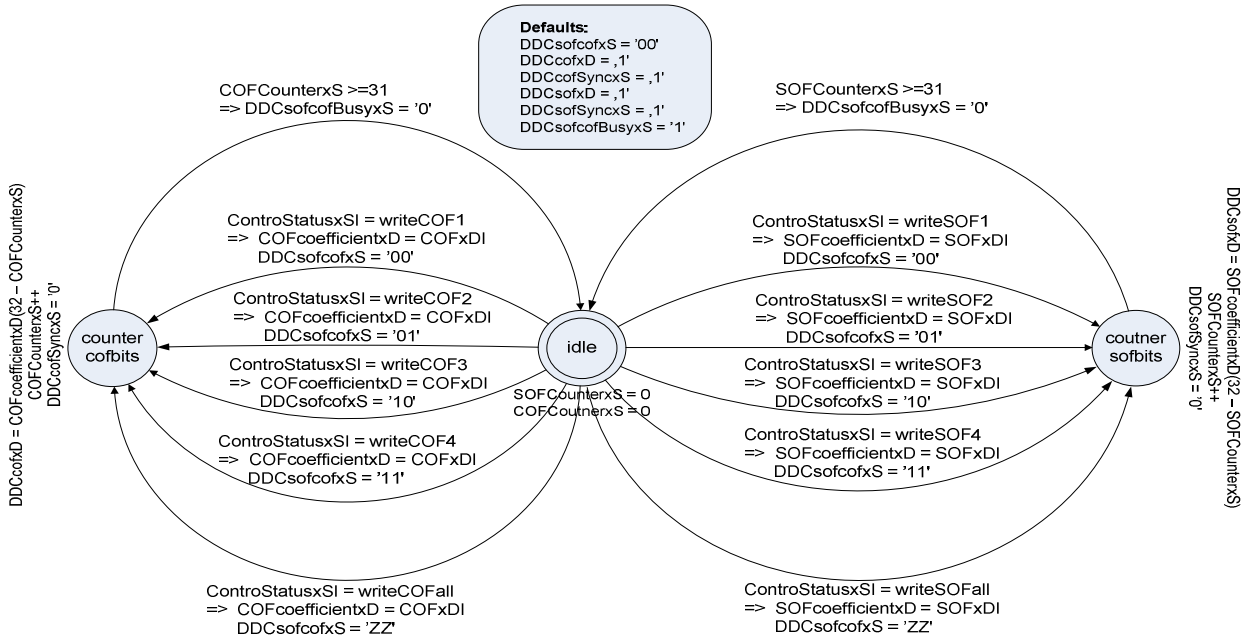


Figure 10: State flow diagram of COF/SOF serial programming serial outputting the SOF or COF configuration bits and stating the selector configuration of the SOF/COF signal demultiplexer unit depending on the system state *Control-StatexSI*.

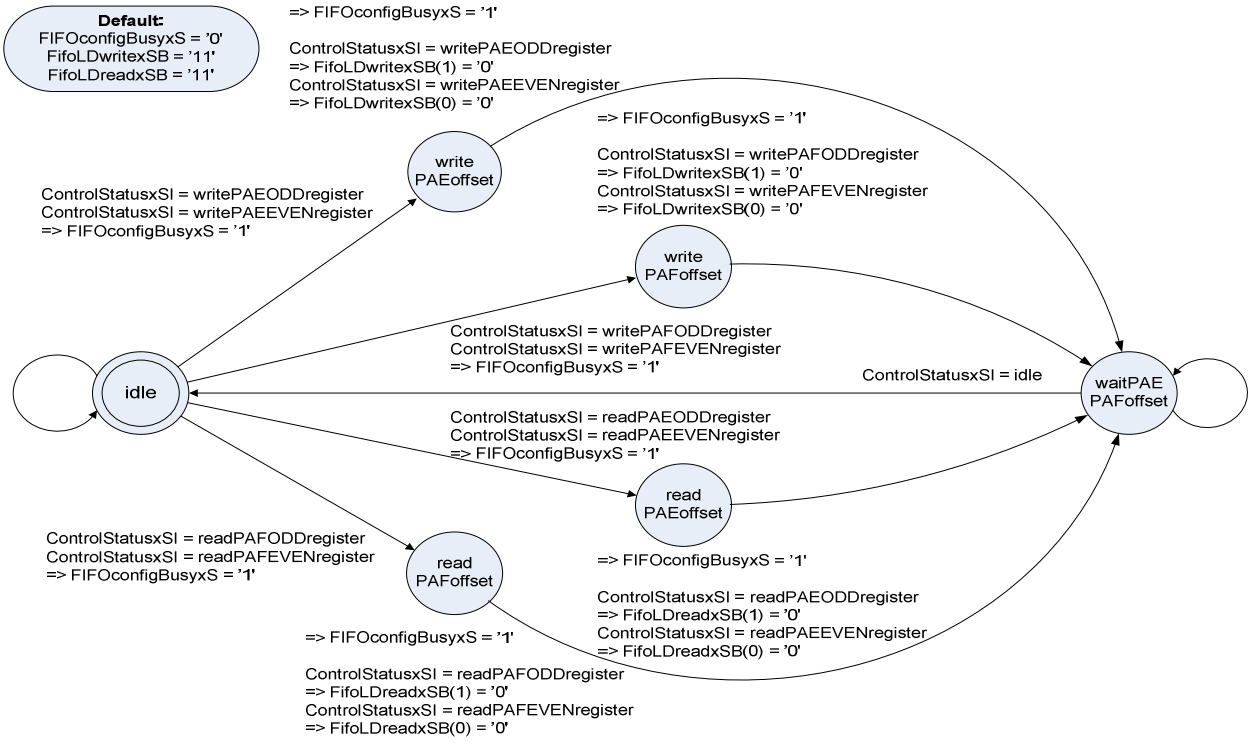


Figure 11: PAE/PAF configuration sequence providing PAEconfigxEO and PAFconfigxEO to VME unit.

Interface Unit FIFO interfaces			
Input	ControlStatusxS	System status	
	FifoEFxSBI[1..0]	FIFO empty flag	odd and even FIFOs
	FifoFFxSBI[1..0]	FIFO full flag	odd and even FIFOs
	FifoPAExSBI[1..0]	FIFO prog almost empty flag	odd and even FIFOs
	FifoPAFxSBI[1..0]	FIFO prog almost full flag	odd and even FIFOs
	DDCDataReadyxSBI	New data ready at the DDC output	
Output	FifoWritexEBO[1..0]	FIFO write CLK enable	odd and even FIFO
	FifoReadxEBO[1..0]	FIFO read CLK enable	odd and even FIFO
	FiFOOutxEBO[1..0]	FIFO output enable	odd and even FIFO
	FifoLDxSBO[1..0]	Enable prog flag offset register	odd and even FIFO
	FifoRSxSBO	Reset FIFO to empty conditions	
	IFstatusxS	Interface status	
	FifoFlagxS[7..0]	FIFO flags (remaining 2 flags are generated on top)	

The *FIFO Interface* unit depicted in Figure 12 collects the FIFO flag data (*FifoEFxSB, FifoFFxSB, FifoPAExSB*, and *FIFOPAFxSB*) as *FifoFlagxSO* and provides them as well as a structure with *DDCDataReadyxSB* as *IFstatusxSO*.

The outputs of the *FIFO Interface* unit build the regular enables *FifoWritexEB*, *FifoReadxEB*, *FifoLDxSB*, and *FifoRSxSB* for FIFO read and write operations depending on the system state *ControlStatusxS*, stored *FifoReadxSP*, and *FifoWritexSP* plus delayed version and as well the FIFO flag information.

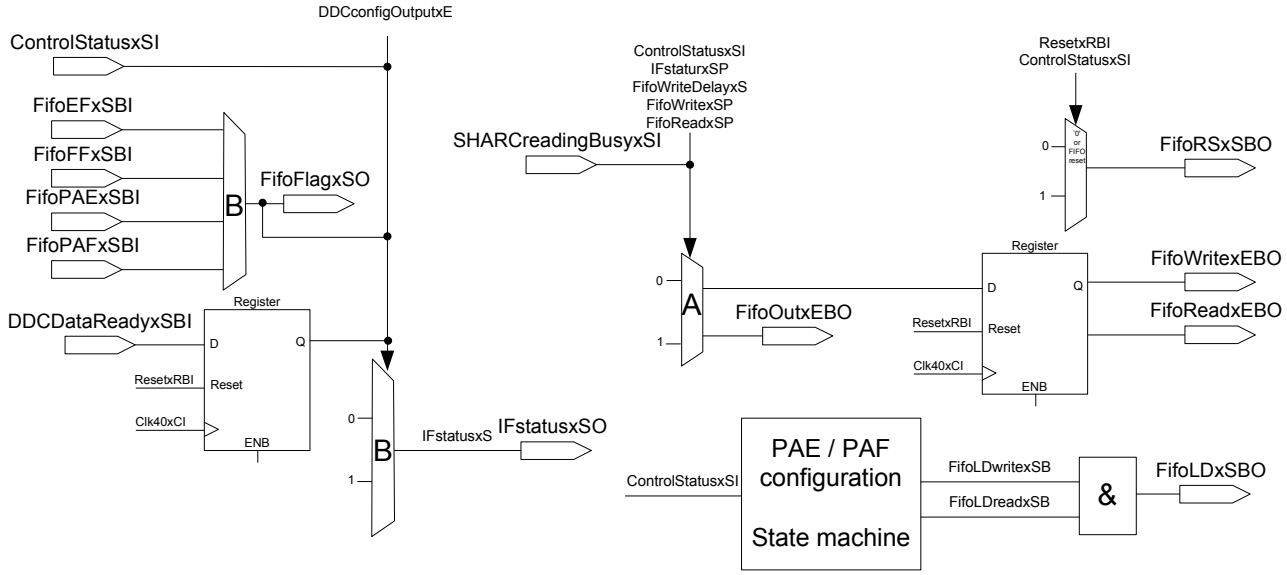


Figure 12: Schematic of *FIFO interface* unit collecting the FIFO flag information and generating the FIFO enable signals for read, write, and reset operations.

Selector A in Figure 12						
ControlStatusxSI	**Register(6)**	**SHARC Reading BusyxSI**	**IFstatusxSP**	**FifoOutxEB**	**FifoWritexSP**	**FifoReadxSP**
Default:		-	-	'00'	'11'	'11'
VME_oddFIFOwrite	'0'	-	-	'00'	'01'	'11'
oddFIFO_VMEread	'0'	-	-	'00'	'11'	'01'
VME_evenFIFOwrite	'0'	-	-	'00'	'10'	'11'
evenFIFO_VMEread	'0'	-	-	'00'	'11'	'10'
VME_bothFIFOwrite	'0'	-	-	'00'	'00'	'11'
bothFIFOreset	'0'	-	-	'11'	'11'	'11'
-	'0'	-	FIFObusyxS = 0 DDCdataReadyxS(0) = '1'	'00'	'10'	'11'
-	'0'	-	FIFObusyxS = 0 DDCdataReadyxS(1) = '1'	'00'	'01'	'11'
-	'0'	'1'	-	'00'	'11'	'00'
writePAEODDregister writePAFODDregister	'1'	-	-	'00'	'01'	'11'
readPAEODDregister readPAFODDregister	'1'	-	-	'00'	'11'	'01'
writePAEEVENregister writePAFEVENregister	'1'	-	-	'00'	'10'	'11'
readPAEEVENregister readPAFEVENregister	'1'	-	-	'00'	'11'	'10'

Selector B in Figure 12			**IFstatusxSP**				
FIFObusyxS	**FIFOEFxS FifoPAExSBI**	**FIFOFFxSBI FifoPAFxSBI**	**FIFOfullxS**	**FIFOemptyxS**	**FIFObusyxS**	**DDCdataReadyxS**	**Fi-foReadxEB**
Init/Default:			'0'	'1'	'0'	'00'	'11'
-	'00'	-	'1'	-	-	not(DDCdataReadyxS)	FifoReadxEB
-	-	'00'	-	'0'	-	not(DDCdataReadyxS)	FifoReadxEB
'1'	'00'	-	-	-	'0'	not(DDCdataReadyxS)	FifoReadxEB
'0'	-	'00'	-	-	1	not(DDCdataReadyxS)	FifoReadxEB

The registers RegisterxDI(4..3) indicates whether the control signal pair FifoEFxSBI-FifoFFxSBI (not(RegisterxDI = '11')) or FifoPAExSBI-FifoPAFxSBI (RegisterxDI = '11') are used.

Interface Unit Selector interface			
Inputs:	ControlStatusxS	System status	
Output	SelectAoutxEBO[1..0]	Select output driver of A interface	odd and even FIFO
	SelectBoutxEBO[1..0]	Select output driver of B interface	odd and even FIFO
	SelectCoutxEBO[1..0]	Select output driver of C interface	odd and even FIFO

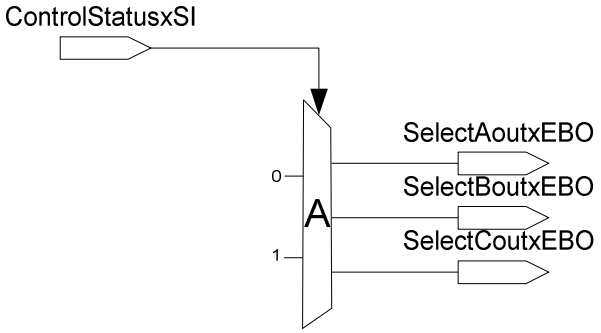


Figure 13: Schematic of the *Selector Interface* unit generating the selector enable signal depending on the system state.

Selector A in Figure 13			
ControlStatusxSI	**SelectAoutxEB**	**SelectBoutxEB**	**SelectCoutxEB**
Default:	'11'	'11'	'11'
VME_oddFIFOwrite	'11'	'11'	'01'
oddFIFO_VMEread	'11'	'01'	'11'
VME_evenFIFOwrite	'11'	'11'	'10'
evenFIFO_VMEread	'11'	'01'	'11'
VME_bothFIFOwrite	'11'	'11'	'00'
bothFIFOreset	'11'	'11'	'11'
writePAEODDregister	'11'	'11'	'01'
writePAFODDregister	'11'	'11'	'01'
readPAEODDregister	'11'	'01'	'11'
readPAFODDregister	'11'	'01'	'11'
writePAEEVENregister	'11'	'11'	'10'
writePAFEVENregister	'11'	'11'	'10'
readPAEEVENregister	'11'	'10'	'11'
readPAFEVENregister	'11'	'10'	'11'

5.5 SHARC Interface Unit

The handshake protocol between C-FPGA and L-FPGA provides the usage of L-FPGA as slave processing the data between FIFO and SHARC. The C-FPGA controls the data transfer process as shown in Figure 14 using request and acknowledge signals (*LDataReadyxSO* and *LDataAckxSI*) and checking the mode dependent on the acquisition of the FIFO.

Sharc Interface Unit interface		
Inputs:	ControlStatusxS	System status
	RegisterxD[15..0]	Command register for extracting monitor functionality
	IFstatusxS	Interface status
	LDataAckxSI	Handshake protocol signal
Outputs:	LDataReadyxSO	Handshake protocol signal
	LResetxSBO	Reset for L-FPGA generated by C-FPGA status condition
	SHARCreadingBusyxS	Blocking while SHARC processing

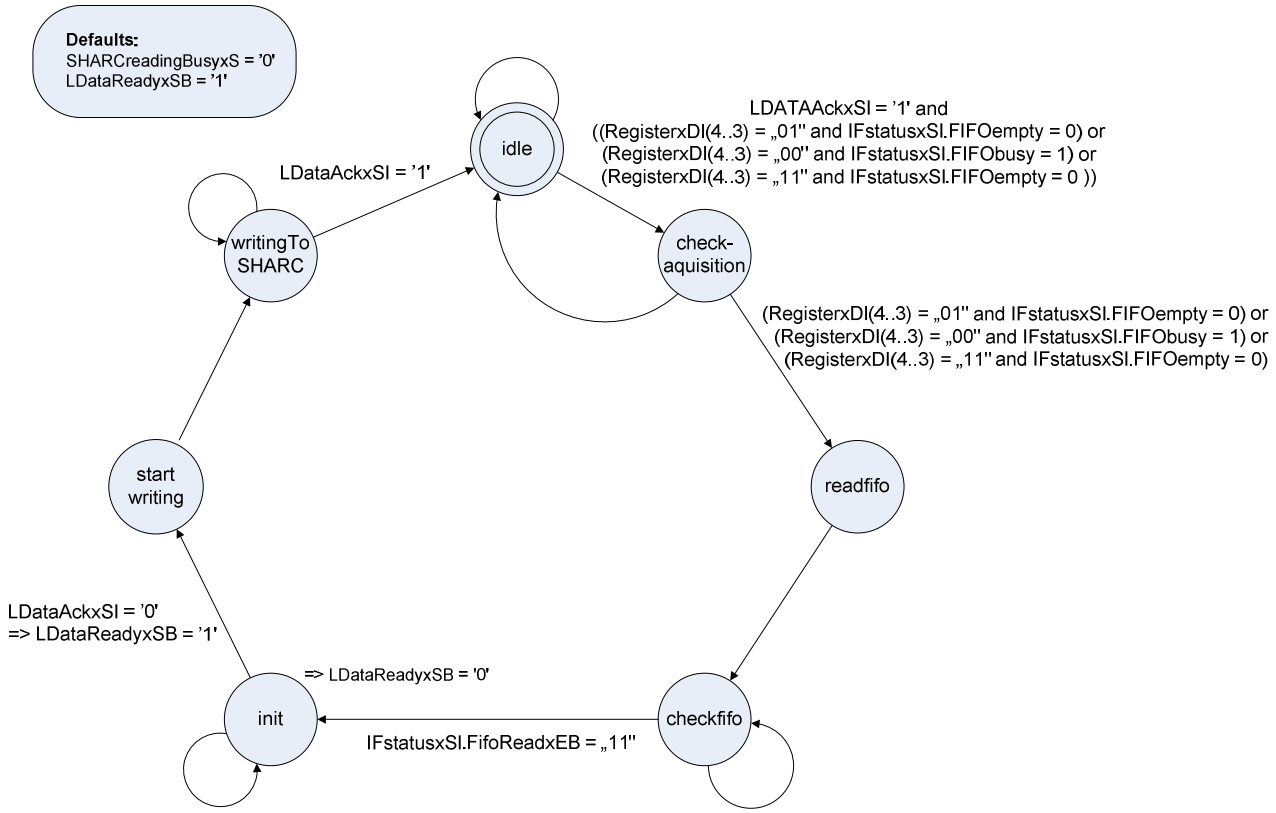


Figure 14: State flow diagram of SHARC interface status with the different acquisition mode dependencies, generating the request signal *LDataReadyxSO* and waiting for an acknowledgment *LDataAckxSI.*

5.6 Monitor Unit

Control state and alert information are generated to monitor the system on the front panel.

Monitor Unit interface		
Inputs:	RegisterxS[15..0]	Command register for extracting monitor functionality
	ControlStatusxS	System status
Outputs:	MonitorDDCsamplexSO	DDCdataReadyxSI[1..0] both active
	MonitorLinkPortxSO	LDataReadyxSI inactive
	MonitorGatexSO	Gate inactive
	MonitorVMEaccessxSO	BoardSelectxS
	MonitorErrorxSO	Command register[5] and [0] inacttive

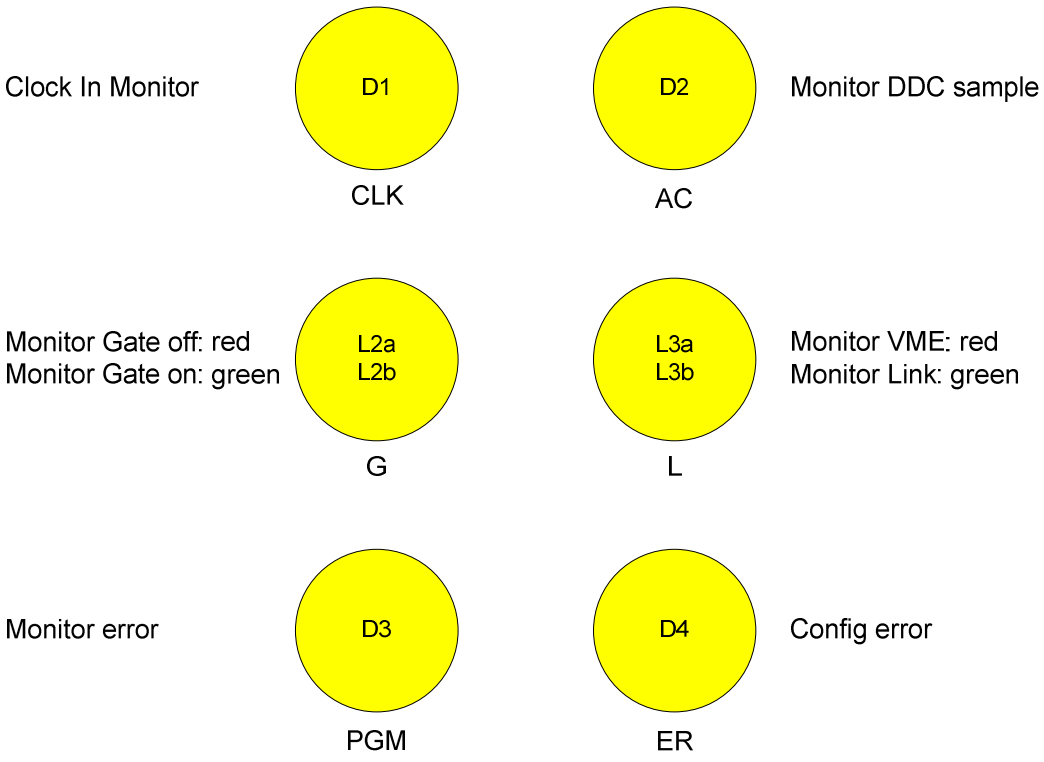


Figure 15: Monitor LED configuration: The six LED show a running clock, active DDC samples, VME access, link port, the gate status, and errors of the system status (command register Link and VME) and the configuration.

The *Gate* and the *VME/Link* LED are two colour LED both offering two channels. In particular the *VME/Link* monitor occupy each a channel while the *Gate* monitor shows its two states. The ordering of this middle LED couple might be mirrored on some hardware boards.

5.7 Gate Synchronization Unit

The external gate signal triggers the acquisition depending on the FIFO status and flag information. The unit generates the dedicated gate signals for internal DDC acquisition *DDCGatexS* and the external forwarded *GatexS* signal.

Gate Synchronization Unit interface		
Inputs:	RegisterxS[15..0]	Command register for mode settings
	DDCDataReadyxSBI	DDC status signal
	FifoFlagxS[7..0]	FIFO flags (remaining 2 flags are generated on top)
	IFstatusxS	Interface status
	GateInxSI	External trigger
Outputs:	GatexSBO	Generated trigger signal
	DDCgatexS	Internal trigger for DDC acquisition

The *Gate Synchronization* unit provides four gate signals depending on *RegisterxDI(2..1)* and a inactive *FIFObusyDelayxS:*

- Disabled gate with *GatexSO* '1'.
- Constant enabled gate with *GatexSO* '0'
- *TriggerGatexSP*: A falling edge of the *GateInxSI* signal generates a pulse on the *TriggerxS* signal. This pulse is brings the *TriggerGatexSP* flip-flip high and the *GatexSO* remains low. A deactivation requires a mode '10' deactivation.
- *DDCgatexSP* is stored high after *TriggerxS* occurred and reset depending on the *IFstatusxSI* and its delayed signals.

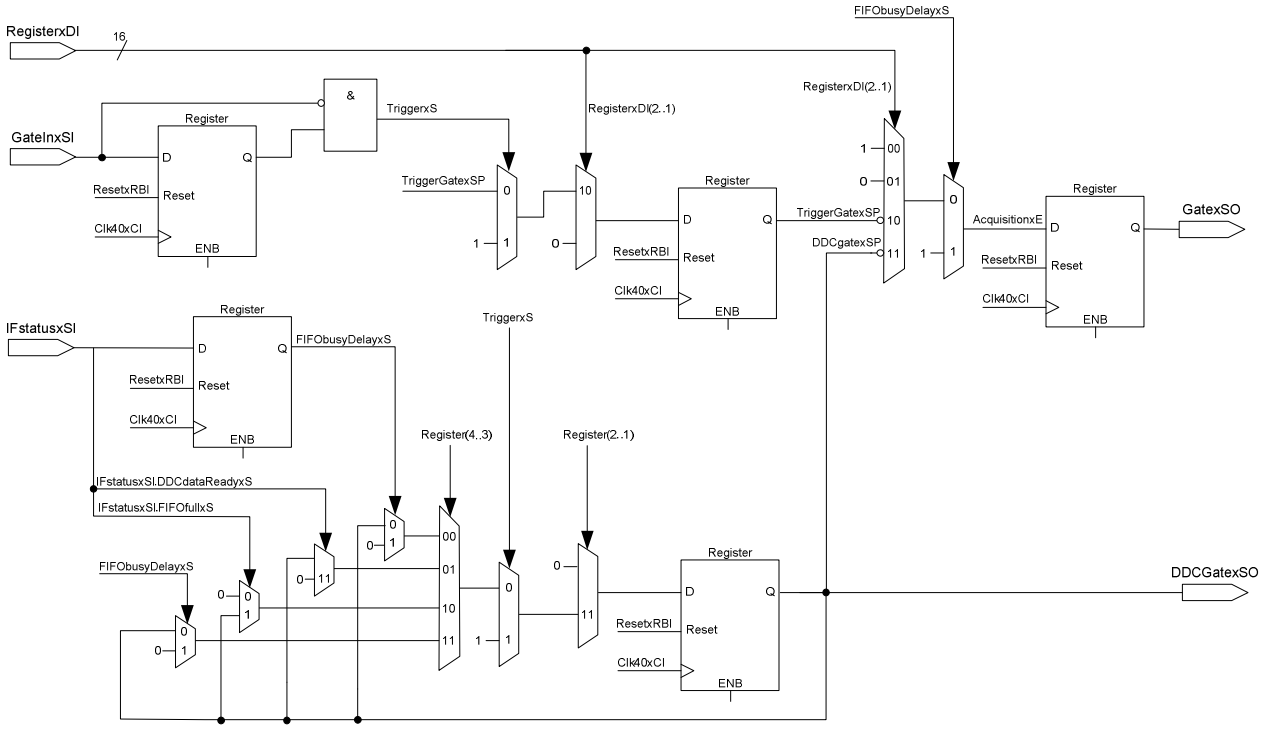


Figure 16: Schematic of *Gate Synchronization* unit generating mode dependent trigger and gate signals.

6. Link FPGA (L-FPGA)

The L-FPGA forwards the parallel received data from the FIFO to the SHARC link. 64 bit vectors *DataxD* are sent in nibbles *LxD* of 4 bits compiled as two data packages of 8 nibbles each to the SHARC drivers. In the control unit the protocol to the C-FPGA and as well the control data to the SHARC controller has to be processed.

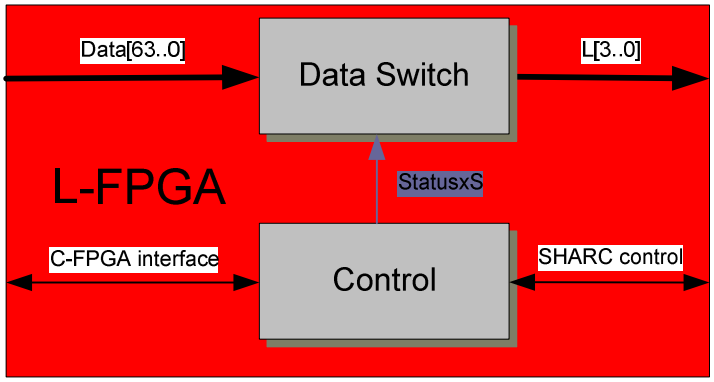


Figure 17: Main units of L-FPGA: The data switch between FIFO and SHARC is a slave of the C-FPGA and locally operated by a control unit

6.1 Control Unit

The unit interfaces with the C-FPGA with a hand-shake and provides the SHARC control line to the DSP:

- With *LDataReadyxSBI* going low the C-FPGA indicates that data is ready to be sent to the SHARC link.
- The *LDataAckxSO* is low while the LFPAG is sending data to the SHARC and high indicating idle or waiting for the link getting ready.
- On the SHARC link side a high *LackxSBI* shows a ready link.
- The data transfer is accompanied by a clock pulses on *L1xCO* for each nibble.
- The signal *LTxSO*, *LRxSBO* , *LackTxSO*, and *LackRxSBO* are kept at constant value for the implemented operations.

Control Unit interface		
Input:	LResetxSBI	Reset provided by C-FPGA
	LDataReadyxSBI	Ready signal from C-FPGA based on the operation state
	LackxSBI	SHARC acknowledge
Output:	LDataAckxSO	Data transfer acknowledgment
	L1xCO	Clock in SHARC interface
	LTxSO	(constant high)
	LRxSBO	(constant low)
	LackTxSO	(constant low)
	LackRxSBO	(constant high)

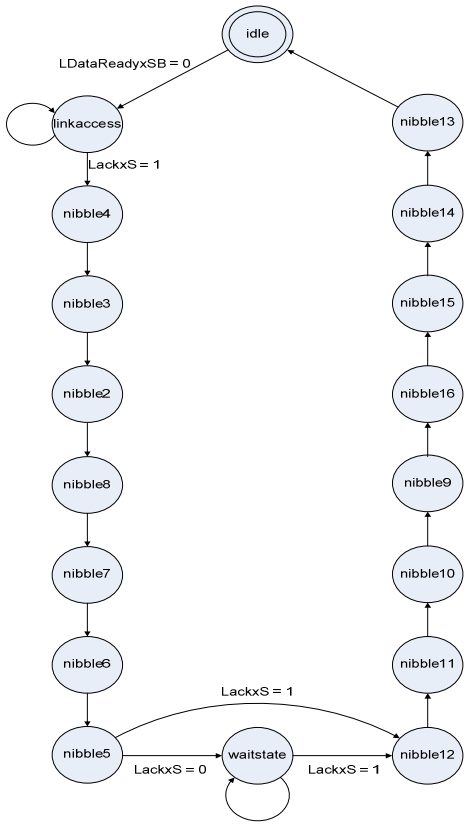


Figure 18: State flow diagram of LFPGA control unit.

The control flow of the LFPGA follows the protocol of the SHARC data structure. The 64 bit data word received by the DDC is provided nibble-by-nibble to the SHARC interface driver.

The input control signals are fed into flipflops: *LDataReadyxSB* is the registered input signal *LDataReadyxSBI*, *LackxS* is the registered input signal *LackxSI*.

Starting in the idle state, the *LDataReadyxSB* signal is required to be low so that the link is expected to become ready (*LackxS* high). After passing these two conditions nibble4 is the first data nibble sent to the SHARC followed by nibbles 3, 2, 1, 8, 7, and 5. Those eight nibbles build the first half.

Depending on the *LackxS*, the system holds in a wait state, or on *LackxS* high continues with nibble12 followed by nibbles 11, 10, 9, 16, 15, 14, and 13.

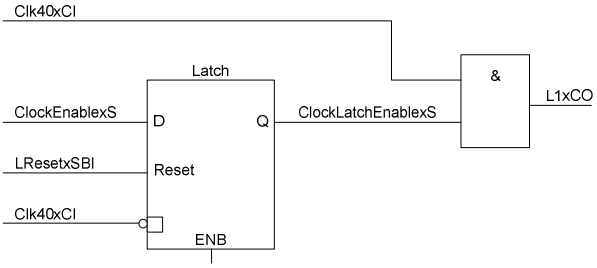


Figure 19: Latch based clock gate circuit.

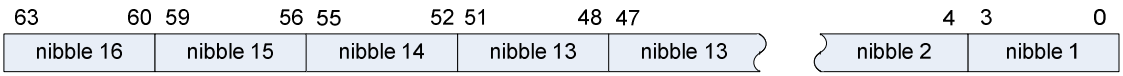
In any but the *idle, linkaccess*, and *wait state* a single clock pulse is generated out of a clock gate as show in Figure 18. The clock gate consists of clock latch active on the low clock phase and a consecutive NAND gate combining the regular clock *Clk40xC* and the generated *ClockLatchEnablexS* to the *L1xCO*.

6.2 Data Switch Unit

The L-FPGA operates as pure data switch interfacing between FIFO and SHARC processor.

Data Switch Unit interface		
Input:	DataxDI	FIFO data bus
Output:	LxDO[3..0]	Data nibble on SHARC interface

In the control state *link access*, the input *LDataxDI* is stored in the local registers *LDataxD*. Depending on the state the corresponding nibble is switched to the output. Nibble 1 labels the least-significant four bits (bits 3 down to bit 0) while nibble 16 indicates the most significant bits (bit 63 down to 60) of *LDataxD*.



7. System Configuration

7.1 Register Map

Writing and Reading to and from the Register Map is possible using address *LAxDI* and data *LDxDI* bus with control signal *RWxSI.*

VME offset	Bit	Description
0x0000	All	Odd DDC FIFO reading or writing (DDC1 Lword, DDC3 Hword)
0x0004	All	Even DDC FIFO reading or writing (DDC2 Lword, DDC4 Hword)
0x0008	All	Writing to all FIFO registers (Lword DDC1/DDC2, Hword DDC3/DD4) in parallel. A reading is only possible in D64 VME mode (currently not used).
0x000C	All	FIFO reset
0x0010	0 [2..1] [4..3] 5 6 7 [9..8] [14..10] 15	VME access enable (data transmission mode C) Gated acquisiton mode (data transmission mode A) FIFO forwarding mode Link port enable FIFO access not used DDC data format not used Board reset
0x0020	All	FIFO status flags
0x0040	All	Static DDC registers
0x0080	All	Global address and version number
0x0100	[3..0]	Interrupt status register: Prioritization as IRQ vector IRQ 4 status flag (lowest priority) IRQ 3 status flag IRQ 2 status flag IRQ 1 status flag (highest priority)
0x0104	[9..0]	X1 register, IRQ flags for FIFO status register
0x0108	[9..0]	M1 register, masking flag (0 ignored, 1 masked/enabled)
0x10C	[31..0]	V1 register, programmable IRQ vector
0x0110		not used
0x0114	[9..0]	X2 register, IRQ flags for FIFO status register
0x0118	[9..0]	M2 register, masking flag (0 ignored, 1 masked/enabled)
0x011C	[31..0]	V2 register, programmable IRQ vector
0x0120		not used
0x0124	[9..0]	X3 register, IRQ flags for FIFO status register
0x0128	[9..0]	M3 register, masking flag (0 ignored, 1 masked/enabled)
0x012C	[31..0]	V3 register, programmable IRQ vector

VME offset	Bit	Description
0x0130		not used
0x0134	[9..0]	X4 register, IRQ flags for FIFO status register
0x0138	[9..0]	M4 register, masking flag (0 ignored, 1 masked/enabled)
0x013C	[31..0]	V4 register, programmable IRQ vector
0x0140	[3..0]	Interropt Enable register
0x0200 0x0204 0x0208 0x020C 0x020F	all all all all all	DDC1 sample offset frequency SOF register DDC2 sample offset frequency SOF register DDC3 sample offset frequency SOF register DDC4 sample offset frequency SOF register All DDC SOF loading
0x0400 0x0404 0x0408 0x040C 0x040F	all all all all all	DDC carrier offset frequency COF register DDC carrier offset frequency COF register DDC carrier offset frequency COF register DDC carrier offset frequency COF register All DDC COF loading
0x0800 0x0804 0x0808 0x080C	all all all all	Odd FIFO alternating PAE/PAF offset register (Programmable Almost Empty / Full flag) Odd FIFO alternating PAE/PAF offset register Even FIFO alternating PAE/PAF offset register (Programmable Almost Empty / Full flag) Even FIFO alternating PAE/PAF offset register
0x1000 0x2000 0x4000 0x8000 0xF000	all all all all all	DDC1 parameters (writing to DDC requires VME suspending) DDC2 parameters (writing to DDC requires VME suspending) DDC3 parameters (writing to DDC requires VME suspending) DDC4 parameters (writing to DDC requires VME suspending) All DDC parameters (writing to DDC requires VME suspending) (delay is defined on the DDC configuration duration on the C-BUS)

7.2 Command Registers (RegisterxD, Address 0x0010)

Nr.	Modes depending on the command registers	Ref.
8.1	**VME Access Enable (RegisterxD[0] data transmission mode C)** VME access for writing to FIFO: 0 Disable FIFO VME access 1 Enable VME access	[15]
8.2	**Gated Acquisition Modes (RegisterxD[2..1], data transmission mode A)** The DDC data acquisition is processed in 4 defined modes triggered by GATE_IN and defining GATE. [00] Disable acquisition [01] Continuous acquisition: Trigger not relevant [10] Continuous triggered acquisition: Trigger signal inits the acquisition. [11] Gated acquisition: Data acquisition after every trigger synchronously.	[15]
8.3	**FIFO Forwarding Modes (RegisterxD[4..3])** The FIFO data is forwarded given by 4 modes: [00] FIFO snapshot *(TURN-BY-TURN)*: After the FIFO is full the data acquisition is suspended while the data is sent to the DSP link port. The data acquisition may restart (according to the Gate acquisition mode) but not earlier than the FIFO is empty. [01] FIFO buffering *(CLOSED-ORBIT)*: FIFO writing and reading is processed simultaneously. If the FIFO is empty (EF) the data is immediately sent to the link port otherwise buffered in the FIFO. In case the FIFO is full (FF) the acquisition is suspended until the FIFO is empty (EF). [10] FIFO free run: If the FIFO is not full (FF), the data is stored in the FIFO. The forwarding is dependent on the output enables *(not implemented in the current version).* [11] FIFO flag mode: Same suspending as FIFO snapshot mode, in addition using PAE and PAF depth *(not implemented in the current version).*	[15]
8.4	**Link Port Enable (RegisterxD[5] L-FPGA configuration interface)** 0 Disable link port 1 Enable link port	[15]
8.5	**FIFO Access (RegisterxD[6], FIFO configuration interface)** 0 FIFO stack access (regular operation) 1 PAE/PAF offset parameter configuration access	[15]

Nr.	Modes depending on the command registers	Ref.
8.6	**DDC Data Format (RegisterxD[9..8], DDC configuration interface)** The DDC data can be accessed in 4 different formats: [00] Acquire only magnitude data [01] Acquire only phase data [10] not used [11] Acquire magnitude and phase data	[15]

7.3 FIFO Status Flags (FIFOflagxS)

Nr.	Flags Depending on the FIFO States	Ref.
9.1	**FIFO Status Flags (address 0x0020, FIFO status interface)** DDC parameters shall be configured as static control. 0 ODD FIFO Empty Flag (EF) 1 ODD FIFO Programmable Empty Flag (PAE) 2 ODD FIFO Programmable Full Flag (PAF) 3 Odd FIFO Full Flag (FF) 4 Even FIFO Empty Flag (EF) 5 Even FIFO Programmable Empty Flag (PAE) 6 Even FIFO Programmable Full Flag (PAF) 7 Even FIFO Full Flag (FF) 8 Gate Monitor 9 FIFO busy flag 10 Link data acknowledge	[15]

7.4 SOF and COF Functionality

Nr.	SOF and COF Functionality to be Implemented in the Interface Unit	Ref.
9.1	**Task of COF and SOF** ▪ The Carrier Offset Frequency is loaded into the Carrier NCO using a serial input pin. Offset may use up to 32 bits. ▪ The Re-Sampler Offset Frequency loads the offset frequency into the Re-Sampler NCO with up to 32 bits	[9]

Nr.	SOF and COF Functionality to be Implemented in the Interface Unit	Ref.
9.2	**Operation of COF and SOF programming** ■ Other programming of the DDC is suspended during SOF and COF programming. ■ The COF sync is asserted one clock before the MSB of the offset frequency word is applied. ■ The SOF sync is asserted one clock before the MSB of the offset frequency word is applied.	[9]

7.5 IRQ Routine

Nr.	IRQ Functionality and Operation	Ref.
10.1	**Task of IRQ routine** ■ The IRQ are activated with the FIFO flags and the X[1..4] register and presented with the corresponding identification vector V[1..4] on the VME data bus. ■ Both FIFOflagsxS[9..0] and X[1..4] register are masked with the four M[1..4] and the IRQ are enabled with IE[1..4]. Using that, the FN[1..4] register indicates the activated IRQ number. ■ Given this IRQ number the IRQ identification vector V[1..4], the FN[1..4], or one of the IRQ register (X[1..4], M[1..4], or V[1..4]) are addressable and read over the VME bus. ■ Initial state is all IRQ deactivated. ■ The identification vector V[1..4] are even values in the range 0 to 255. ■ The lowest IRQ number corresponds with the highest IRQ priority.	[5]
10.2	**IRQ Operation** ■ If the FIFO status register equals X[m] (masked with M[m]) the IRQ bit of the FN[m] register is set. In case one IRQ bit is triggered *lirq* is asserted low. ■ When *lden* is received V[m] of the IRQ with highest priority is put onto the data bus. ■ If more IRQ are triggered simultaneously, the IRQ with the lowest IRQ number has the highest priority ■ The *lirq* is released upon interrupt acknowledge with *lden* and the corresponding IRQ is cleared while writing the FN register. ■ If not all the active IRQ are cleared with the aforementioned write operation or new IRQ are triggered, the *lirq* signal is triggered and the remaining interrupts are treated based on the IRQ priority.	[5]

7.6 Static Configuration

<table>
<tr><th>Nr.</th><th>Dedicated Functionality based on Static Configuration</th><th>Ref.</th></tr>
<tr><td>11.1</td><td>Reset
■ The FIFO can be reset with writing on address 0x000C (FIFO_RS)
■ The FPGA and the VME interface are reset on RegisterxD[15] @ address 0x0010. The FIFO values are unchanged.
Init modes are all modes 0 (VME access mode disabled, disabled acquisition, FIFO forwarding mode 0, disabled link port)</td><td>[15]</td></tr>
<tr><td>11.2</td><td>Static DDC Registers (address 0x0040, DDC configuration interface, DDC port signals, write-only)
DDC parameters shall be configured as static control stored via LD bus in the onboard registers clocked with the gated clock VDATACLK. The configuration data is provided by the VME so that AGCNSEL, GAINADJ, SEL for DDC[1..4] are stored in external FF clocked with VDATACLK.
0 DDC1 AGCNSEL AGC loop gain select
[3..1] DDC1 GAINADJ[2..0] Gain offset “000” = 0dB / “111” = 42 dB
4 n/c
[7..5] DDC1 SEL[2..0] Data format representation at output AOUT & BOUT
8 DDC2 AGCNSEL AGC loop gain select
[11..9] DDC2 GAINADJ[2..0] Gain offset “000” = 0dB / “111” = 42 dB
12 n/c
[15..13] DDC2 SEL[2..0] Data format representation at output AOUT & BOUT
16 DDC3 AGCNSEL AGC loop gain select
[19..17] DDC3 GAINADJ[2..0] Gain offset “000” = 0dB / “111” = 42 dB
20 n/c
[23..21] DDC3 SEL[2..0] Data format representation at output AOUT & BOUT
24 DDC4 AGCNSEL AGC loop gain select
[27..25] DDC4 GAINADJ[2..0] Gain offset “000” = 0dB / “111” = 42 dB
28 n/c
[31..29] DDC4 SEL[2..0] Data format representation at output AOUT & BOUT

DDC Output Data Type
The output data type of the DDC units can be selected over the VME bus while writing the corresponding DDC SEL[2..0] to address 0x0040 and enabling the gated clock VDATACLK.
SEL[2..0] | Output data type
000 I data
001 Q data
010 Magnitude
011 Phase
100 Frequency
101 unused
110 Memory status
111 Incrementing always to the next set</td><td>[15]</td></tr>
</table>

Nr.	Dedicated Functionality based on Static Configuration	Ref.
11.3	**Programming FIFO offset register PAE, PAF** PAE and PAF determine the distance from Empty/Full to Almost Empty/Full which status are flagged by the corresponding flag. Reading and writing is enabled with FIFO_ODD/EVEN_LD, FIFO_ODD/EVEN_REN, and FIFO_ODD/EVEN_WEN active low. ▪ PAE and PAF are selective on the address 0x0000, 0x0004, or 0x0008 and depending on the enable by the command register @ 0x0010. ▪ PAE is written on the first active WCLK edge, PAF on the second, reading operated for the PAE on the second and FAF on the third active clock edge after LD, WEN respectively REN.	[8]
11.4	**Global Address and Version Number (address 0x0080)** [31..24] "Q" 23 "0" [22.18] Slot number (Global Address) [17..16] "0" [15..0] Version number	[15]

8. Pin-out information

8.1 Pin location C-FPGA version V11

Pin Name/ Usage C-FPGA V11	Location	Dir.	I/O Standard	Voltage	User Defined
TCK	1	input			
CONF_DONE	2	bidir			
nCEO	3	output			
TDO	4	output			
VCC_INT	5	power		3.3V	
la[16]	6	input	LVTTL/LVCMOS		Y
addr[2]	7	output	LVTTL/LVCMOS		Y
addr[1]	8	output	LVTTL/LVCMOS		Y
addr[0]	9	output	LVTTL/LVCMOS		Y
GND_INT	10	gnd			
promclock	11	input	LVTTL/LVCMOS		Y
c[0]	12	output	LVTTL/LVCMOS		Y
c[1]	13	output	LVTTL/LVCMOS		Y
c[2]	14	output	LVTTL/LVCMOS		Y
c[3]	15	output	LVTTL/LVCMOS		Y
VCC_INT	16	power		3.3V	
c[4]	17	output	LVTTL/LVCMOS		Y
c[5]	18	output	LVTTL/LVCMOS		Y
c[6]	19	output	LVTTL/LVCMOS		Y
c[7]	20	output	LVTTL/LVCMOS		Y
/ddcw	21	output	LVTTL/LVCMOS		Y
GND_INT	22	gnd			
prom_oe	23	input	LVTTL/LVCMOS		Y
/ddcr	24	output	LVTTL/LVCMOS		Y
ddc4_/oe	25	output	LVTTL/LVCMOS		Y
GND*	26				
VCC_INT	27	power		3.3V	
ddc4_w/r	28	output	LVTTL/LVCMOS		Y
ddc3_/oe	29	output	LVTTL/LVCMOS		Y
ddc3_w/r	30	output	LVTTL/LVCMOS		Y
ddc2_/oe	31	output	LVTTL/LVCMOS		Y
GND_INT	32	gnd			
ddc2_w/r	33	output	LVTTL/LVCMOS		Y
ddc1_/oe	34	output	LVTTL/LVCMOS		Y
ddc1_w/r	35	output	LVTTL/LVCMOS		Y
selb	36	output	LVTTL/LVCMOS		Y
VCC_INT	37	power		3.3V	
sela	38	output	LVTTL/LVCMOS		Y
selc	39	output	LVTTL/LVCMOS		Y
fifo_odd_/ld	40	output	LVTTL/LVCMOS		Y
fifo_odd_/ren	41	output	LVTTL/LVCMOS		Y
GND_INT	42	gnd			
fifo_odd_/wen	43	output	LVTTL/LVCMOS		Y
fifo_odd_/oe	44	output	LVTTL/LVCMOS		Y
/oec_odd	45	output	LVTTL/LVCMOS		Y
/oeb_odd	46	output	LVTTL/LVCMOS		Y
VCC_INT	47	power		3.3V	
/oea_odd	48	output	LVTTL/LVCMOS		Y
fifo_even_/ld	49	output	LVTTL/LVCMOS		Y
fifo_even_/ren	50	output	LVTTL/LVCMOS		Y
fifo_even_/wen	51	output	LVTTL/LVCMOS		Y
GND_INT	52	gnd			
fifo_even_/oe	53	output	LVTTL/LVCMOS		Y
/oec_even	54	output	LVTTL/LVCMOS		Y
/oeb_even	55	output	LVTTL/LVCMOS		Y
/oea_even	56	output	LVTTL/LVCMOS		Y
VCC_INT	57	power		3.3V	
TMS	58	input			
TRST	59	input			
nSTATUS	60	bidir			
fifo_/rs	61	output	LVTTL/LVCMOS		Y
GND*	62				
GND*	63				
ddc_sof	64	output	LVTTL/LVCMOS		Y
ddc_sofsync	65	output	LVTTL/LVCMOS		Y
ddc_cof	66	output	LVTTL/LVCMOS		Y
ddc_cofsync	67	output	LVTTL/LVCMOS		Y
ddc_sofcof_sel[1]	68	output	LVTTL/LVCMOS		Y
GND_INT	69	gnd			
ddc_sofcof_sel[0]	70	output	LVTTL/LVCMOS		Y
gate_in	71	input	LVTTL/LVCMOS		Y
/gate	72	output	LVTTL/LVCMOS		Y
/boardsel	73	input	LVTTL/LVCMOS		Y
vdataclk	74	output	LVTTL/LVCMOS		Y
monitor_error	75	output	LVTTL/LVCMOS		Y
monitor_gate	76	output	LVTTL/LVCMOS		Y
VCC_INT	77	power		3.3V	
monitor_link_port	78	output	LVTTL/LVCMOS		Y
monitor_vme_access	79	output	LVTTL/LVCMOS		Y
monitor_ddcs_sample	80	output	LVTTL/LVCMOS		Y
ld[31]	81	bidir	LVTTL/LVCMOS		Y
ld[30]	82	bidir	LVTTL/LVCMOS		Y
ld[29]	83	bidir	LVTTL/LVCMOS		Y
ld[28]	84	bidir	LVTTL/LVCMOS		Y
GND_INT	85	gnd			
ld[27]	86	bidir	LVTTL/LVCMOS		Y
ld[26]	87	bidir	LVTTL/LVCMOS		Y

Pin Name/ Usage C-FPGA V11	Loca-tion	Dir.	I/O Standard	Voltage	User Defined
ld[25]	88	bidir	LVTTL/LVCMOS		Y
VCC_INT	89	power		3.3V	
fifo_half_full	90	input	LVTTL/LVCMOS		Y
clock80	91	input	LVTTL/LVCMOS		Y
fifo_empty	92	input	LVTTL/LVCMOS		Y
GND_INT	93	gnd			
ld[24]	94	bidir	LVTTL/LVCMOS		Y
la[31]	95	input	LVTTL/LVCMOS		Y
VCC_INT	96	power		3.3V	
la[30]	97	input	LVTTL/LVCMOS		Y
la[29]	98	input	LVTTL/LVCMOS		Y
la[28]	99	input	LVTTL/LVCMOS		Y
la[27]	100	input	LVTTL/LVCMOS		Y
la[26]	101	input	LVTTL/LVCMOS		Y
la[25]	102	input	LVTTL/LVCMOS		Y
la[24]	103	input	LVTTL/LVCMOS		Y
GND_INT	104	gnd			
ld[23]	105	bidir	LVTTL/LVCMOS		Y
ld[22]	106	bidir	LVTTL/LVCMOS		Y
ld[21]	107	bidir	LVTTL/LVCMOS		Y
ld[20]	108	bidir	LVTTL/LVCMOS		Y
ld[19]	109	bidir	LVTTL/LVCMOS		Y
ld[18]	110	bidir	LVTTL/LVCMOS		Y
ld[17]	111	bidir	LVTTL/LVCMOS		Y
VCC_INT	112	power		3.3V	
ld[16]	113	bidir	LVTTL/LVCMOS		Y
la[23]	114	input	LVTTL/LVCMOS		Y
la[22]	115	input	LVTTL/LVCMOS		Y
la[21]	116	input	LVTTL/LVCMOS		Y
la[20]	117	input	LVTTL/LVCMOS		Y
la[19]	118	input	LVTTL/LVCMOS		Y
la[18]	119	input	LVTTL/LVCMOS		Y
la[17]	120	input	LVTTL/LVCMOS		Y
nCONFIG	121	input			
VCC_INT	122	power		3.3V	
MSEL1	123	input			
MSEL0	124	input			
GND_INT	125	gnd			
ddc_even_/datardy	126	input	LVTTL/LVCMOS		Y
ddc_odd_/datardy	127	input	LVTTL/LVCMOS		Y
/softreset	128	output	LVTTL/LVCMOS		Y
GND*	129				
VCC_INT	130	power		3.3V	
ud_clkab	131	input	LVTTL/LVCMOS		Y
ud_clkba	132	input	LVTTL/LVCMOS		Y
/ud_oeab	133	input	LVTTL/LVCMOS		Y
ud_sab	134	input	LVTTL/LVCMOS		Y
GND_INT	135	gnd			

Pin Name/ Usage C-FPGA V11	Loca-tion	Dir.	I/O Standard	Voltage	User Defined
ud_sba	136	input	LVTTL/LVCMOS		Y
/ud_oeba	137	input	LVTTL/LVCMOS		Y
/lack	138	output	LVTTL/LVCMOS		Y
/lirq	139	output	LVTTL/LVCMOS		Y
VCC_INT	140	power		3.3V	
region[0]	141	output	LVTTL/LVCMOS		Y
region[1]	142	output	LVTTL/LVCMOS		Y
region[2]	143	output	LVTTL/LVCMOS		Y
region[3]	144	output	LVTTL/LVCMOS		Y
GND_INT	145	gnd			
dbe[0]	146	input	LVTTL/LVCMOS		Y
dbe[1]	147	input	LVTTL/LVCMOS		Y
dbe[2]	148	input	LVTTL/LVCMOS		Y
dbe[3]	149	input	LVTTL/LVCMOS		Y
VCC_INT	150	power		3.3V	
cs[0]	151	input	LVTTL/LVCMOS		Y
cs[1]	152	input	LVTTL/LVCMOS		Y
cs[2]	153	input	LVTTL/LVCMOS		Y
cs[3]	154	input	LVTTL/LVCMOS		Y
GND_INT	155	gnd			
cs[4]	156	input	LVTTL/LVCMOS		Y
cs[5]	157	input	LVTTL/LVCMOS		Y
d64	158	input	LVTTL/LVCMOS		Y
/denin1	159	input	LVTTL/LVCMOS		Y
VCC_INT	160	power		3.3V	
/denin	161	input	LVTTL/LVCMOS		Y
/lden	162	input	LVTTL/LVCMOS		Y
GND*	163				
GND*	164				
GND_INT	165	gnd			
ladi	166	input	LVTTL/LVCMOS		Y
/aben	167	input	LVTTL/LVCMOS		Y
laen	168	input	LVTTL/LVCMOS		Y
GND*	169				
VCC_INT	170	power		3.3V	
GND*	171				
GND*	172				
GND*	173				
ddc_even_/intrpt	174	input	LVTTL/LVCMOS		Y
ddc_odd_/intrpt	175	input	LVTTL/LVCMOS		Y
GND_INT	176	gnd			
TDI	177	input			
nCE	178	input			
DCLK	179	bidir			
DATA0	180	input			
/l_reset	181	output	LVTTL/LVCMOS		Y
fifo_even_/ef	182	input	LVTTL/LVCMOS		Y
fifo_even_/ff	183	input	LVTTL/LVCMOS		Y

Pin Name/ Usage C-FPGA V11	Loca-tion	Dir.	I/O Standard	Voltage	User Defined
r/w	184	input	LVTTL/LVCMOS		Y
fifo_even_/pae	185	input	LVTTL/LVCMOS		Y
/l_data_ready	186	output	LVTTL/LVCMOS		Y
ddc_even_/oea	187	output	LVTTL/LVCMOS		Y
l_data_ack	188	input	LVTTL/LVCMOS		Y
VCC_INT	189	power		3.3V	
fifo_even_/paf	190	input	LVTTL/LVCMOS		Y
ddc_even_/oeb	191	output	LVTTL/LVCMOS		Y
ddc_odd_/oea	192	output	LVTTL/LVCMOS		Y
ddc_odd_/oeb	193	output	LVTTL/LVCMOS		Y
GND*	194				
ld[7]	195	bidir	LVTTL/LVCMOS		Y
ld[6]	196	bidir	LVTTL/LVCMOS		Y
GND_INT	197	gnd			
ld[5]	198	bidir	LVTTL/LVCMOS		Y
ld[4]	199	bidir	LVTTL/LVCMOS		Y
ld[3]	200	bidir	LVTTL/LVCMOS		Y
ld[2]	201	bidir	LVTTL/LVCMOS		Y
ld[1]	202	bidir	LVTTL/LVCMOS		Y
ld[0]	203	bidir	LVTTL/LVCMOS		Y
la[7]	204	input	LVTTL/LVCMOS		Y
VCC_INT	205	power		3.3V	
la[6]	206	input	LVTTL/LVCMOS		Y
la[5]	207	input	LVTTL/LVCMOS		Y
la[4]	208	input	LVTTL/LVCMOS		Y
/reset	209	input	LVTTL/LVCMOS		Y
fifo_full	210	input	LVTTL/LVCMOS		Y
clock40	211	input	LVTTL/LVCMOS		Y
ddc_clk	212	input	LVTTL/LVCMOS		Y

Pin Name/ Usage C-FPGA V11	Loca-tion	Dir.	I/O Standard	Voltage	User Defined
GND*	213				
la[3]	214	input	LVTTL/LVCMOS		Y
la[2]	215	input	LVTTL/LVCMOS		Y
GND_INT	216	gnd			
la[1]	217	input	LVTTL/LVCMOS		Y
la[0]	218	input	LVTTL/LVCMOS		Y
ld[15]	219	bidir	LVTTL/LVCMOS		Y
ld[14]	220	bidir	LVTTL/LVCMOS		Y
ld[13]	221	bidir	LVTTL/LVCMOS		Y
ld[12]	222	bidir	LVTTL/LVCMOS		Y
ld[11]	223	bidir	LVTTL/LVCMOS		Y
VCC_INT	224	power		3.3V	
ld[10]	225	bidir	LVTTL/LVCMOS		Y
ld[9]	226	bidir	LVTTL/LVCMOS		Y
ld[8]	227	bidir	LVTTL/LVCMOS		Y
la[15]	228	input	LVTTL/LVCMOS		Y
la[14]	229	input	LVTTL/LVCMOS		Y
la[13]	230	input	LVTTL/LVCMOS		Y
la[12]	231	input	LVTTL/LVCMOS		Y
GND_INT	232	gnd			
la[11]	233	input	LVTTL/LVCMOS		Y
la[10]	234	input	LVTTL/LVCMOS		Y
la[9]	235	input	LVTTL/LVCMOS		Y
fifo_odd_/ef	236	input	LVTTL/LVCMOS		Y
la[8]	237	input	LVTTL/LVCMOS		Y
fifo_odd_/pae	238	input	LVTTL/LVCMOS		Y
fifo_odd_/paf	239	input	LVTTL/LVCMOS		Y
fifo_odd_/ff	240	input	LVTTL/LVCMOS		Y

8.2 Pin location C-FPGA version V12

Pin Name/ Usage C-FPGA V12	Location	Dir.	I/O Standard	Voltage	User Defined
TCK	1	input			
CONF_DONE	2	bidir			
nCEO	3	output			
TDO	4	output			
VCC_INT	5	power		3.3V	
la[16]	6	input	LVTTL/LVCMOS		Y
addr[2]	7	output	LVTTL/LVCMOS		Y
addr[1]	8	output	LVTTL/LVCMOS		Y
addr[0]	9	output	LVTTL/LVCMOS		Y
GND_INT	10	gnd			
promclock	11	input	LVTTL/LVCMOS		Y
c[0]	12	output	LVTTL/LVCMOS		Y
c[1]	13	output	LVTTL/LVCMOS		Y
c[2]	14	output	LVTTL/LVCMOS		Y
c[3]	15	output	LVTTL/LVCMOS		Y
VCC_INT	16	power		3.3V	
c[4]	17	output	LVTTL/LVCMOS		Y
c[5]	18	output	LVTTL/LVCMOS		Y
c[6]	19	output	LVTTL/LVCMOS		Y
c[7]	20	output	LVTTL/LVCMOS		Y
/ddcw	21	output	LVTTL/LVCMOS		Y
GND_INT	22	gnd			
prom_oe	23	input	LVTTL/LVCMOS		Y
/ddcr	24	output	LVTTL/LVCMOS		Y
ddc4_/oe	25	output	LVTTL/LVCMOS		Y
GND*	26				
VCC_INT	27	power		3.3V	
ddc4_w/r	28	output	LVTTL/LVCMOS		Y
ddc3_/oe	29	output	LVTTL/LVCMOS		Y
ddc3_w/r	30	output	LVTTL/LVCMOS		Y
ddc2_/oe	31	output	LVTTL/LVCMOS		Y
GND_INT	32	gnd			
ddc2_w/r	33	output	LVTTL/LVCMOS		Y
ddc1_/oe	34	output	LVTTL/LVCMOS		Y
ddc1_w/r	35	output	LVTTL/LVCMOS		Y
selb	36	output	LVTTL/LVCMOS		Y
VCC_INT	37	power		3.3V	
sela	38	output	LVTTL/LVCMOS		Y
selc	39	output	LVTTL/LVCMOS		Y
fifo_odd_/ld	40	output	LVTTL/LVCMOS		Y
fifo_odd_/ren	41	output	LVTTL/LVCMOS		Y
GND_INT	42	gnd			
fifo_odd_/wen	43	output	LVTTL/LVCMOS		Y
fifo_odd_/oe	44	output	LVTTL/LVCMOS		Y
/oec_odd	45	output	LVTTL/LVCMOS		Y
/oeb_odd	46	output	LVTTL/LVCMOS		Y
VCC_INT	47	power		3.3V	
/oea_odd	48	output	LVTTL/LVCMOS		Y
fifo_even_/ld	49	output	LVTTL/LVCMOS		Y
fifo_even_/ren	50	output	LVTTL/LVCMOS		Y
fifo_even_/wen	51	output	LVTTL/LVCMOS		Y
GND_INT	52	gnd			
fifo_even_/oe	53	output	LVTTL/LVCMOS		Y
/oec_even	54	output	LVTTL/LVCMOS		Y
/oeb_even	55	output	LVTTL/LVCMOS		Y
/oea_even	56	output	LVTTL/LVCMOS		Y
VCC_INT	57	power		3.3V	
TMS	58	input			
TRST	59	input			
nSTATUS	60	bidir			
fifo_/rs	61	output	LVTTL/LVCMOS		Y
coder[2]	62	input	LVTTL/LVCMOS		Y
coder[1]	63	input	LVTTL/LVCMOS		Y
ddc_sof	64	output	LVTTL/LVCMOS		Y
ddc_sofsync	65	output	LVTTL/LVCMOS		Y
ddc_cof	66	output	LVTTL/LVCMOS		Y
ddc_cofsync	67	output	LVTTL/LVCMOS		Y
ddc_sofcof_sel[1]	68	output	LVTTL/LVCMOS		Y
GND_INT	69	gnd			
ddc_sofcof_sel[0]	70	output	LVTTL/LVCMOS		Y
gate_in	71	input	LVTTL/LVCMOS		Y
/gate	72	output	LVTTL/LVCMOS		Y
coder[0]	73	input	LVTTL/LVCMOS		Y
vdataclk	74	output	LVTTL/LVCMOS		Y
monitor_error	75	output	LVTTL/LVCMOS		Y
monitor_gate	76	output	LVTTL/LVCMOS		Y
VCC_INT	77	power		3.3V	
monitor_link_port	78	output	LVTTL/LVCMOS		Y
monitor_vme_access	79	output	LVTTL/LVCMOS		Y
monitor_ddcs_sample	80	output	LVTTL/LVCMOS		Y
ld[31]	81	bidir	LVTTL/LVCMOS		Y
ld[30]	82	bidir	LVTTL/LVCMOS		Y
ld[29]	83	bidir	LVTTL/LVCMOS		Y
ld[28]	84	bidir	LVTTL/LVCMOS		Y
GND_INT	85	gnd			
ld[27]	86	bidir	LVTTL/LVCMOS		Y
ld[26]	87	bidir	LVTTL/LVCMOS		Y
ld[25]	88	bidir	LVTTL/LVCMOS		Y
VCC_INT	89	power		3.3V	
fifo_half_full	90	input	LVTTL/LVCMOS		Y

Pin Name/ Usage C-FPGA V12	Loca-tion	Dir.	I/O Standard	Voltage	User Defined
clock80	91	input	LVTTL/LVCMOS		Y
fifo_empty	92	input	LVTTL/LVCMOS		Y
GND_INT	93	gnd			
ld[24]	94	bidir	LVTTL/LVCMOS		Y
la[31]	95	input	LVTTL/LVCMOS		Y
VCC_INT	96	power		3.3V	
la[30]	97	input	LVTTL/LVCMOS		Y
la[29]	98	input	LVTTL/LVCMOS		Y
la[28]	99	input	LVTTL/LVCMOS		Y
la[27]	100	input	LVTTL/LVCMOS		Y
la[26]	101	input	LVTTL/LVCMOS		Y
la[25]	102	input	LVTTL/LVCMOS		Y
la[24]	103	input	LVTTL/LVCMOS		Y
GND_INT	104	gnd			
ld[23]	105	bidir	LVTTL/LVCMOS		Y
ld[22]	106	bidir	LVTTL/LVCMOS		Y
ld[21]	107	bidir	LVTTL/LVCMOS		Y
ld[20]	108	bidir	LVTTL/LVCMOS		Y
ld[19]	109	bidir	LVTTL/LVCMOS		Y
ld[18]	110	bidir	LVTTL/LVCMOS		Y
ld[17]	111	bidir	LVTTL/LVCMOS		Y
VCC_INT	112	power		3.3V	
ld[16]	113	bidir	LVTTL/LVCMOS		Y
la[23]	114	input	LVTTL/LVCMOS		Y
la[22]	115	input	LVTTL/LVCMOS		Y
la[21]	116	input	LVTTL/LVCMOS		Y
la[20]	117	input	LVTTL/LVCMOS		Y
la[19]	118	input	LVTTL/LVCMOS		Y
la[18]	119	input	LVTTL/LVCMOS		Y
la[17]	120	input	LVTTL/LVCMOS		Y
nCONFIG	121	input			
VCC_INT	122	power		3.3V	
MSEL1	123	input			
MSEL0	124	input			
GND_INT	125	gnd			
ddc_even_/datardy	126	input	LVTTL/LVCMOS		Y
ddc_odd_/datardy	127	input	LVTTL/LVCMOS		Y
/softreset	128	output	LVTTL/LVCMOS		Y
coder[3]	129	input	LVTTL/LVCMOS		Y
VCC_INT	130	power		3.3V	
ud_clkab	131	input	LVTTL/LVCMOS		Y
ud_clkba	132	input	LVTTL/LVCMOS		Y
/ud_oeab	133	input	LVTTL/LVCMOS		Y
ud_sab	134	input	LVTTL/LVCMOS		Y
GND_INT	135	gnd			
ud_sba	136	input	LVTTL/LVCMOS		Y
/ud_oeba	137	input	LVTTL/LVCMOS		Y
/lack	138	output	LVTTL/LVCMOS		Y

Pin Name/ Usage C-FPGA V12	Loca-tion	Dir.	I/O Standard	Voltage	User Defined
/lirq	139	output	LVTTL/LVCMOS		Y
VCC_INT	140	power		3.3V	
region[0]	141	output	LVTTL/LVCMOS		Y
region[1]	142	output	LVTTL/LVCMOS		Y
region[2]	143	output	LVTTL/LVCMOS		Y
region[3]	144	output	LVTTL/LVCMOS		Y
GND_INT	145	gnd			
dbe[0]	146	input	LVTTL/LVCMOS		Y
dbe[1]	147	input	LVTTL/LVCMOS		Y
dbe[2]	148	input	LVTTL/LVCMOS		Y
dbe[3]	149	input	LVTTL/LVCMOS		Y
VCC_INT	150	power		3.3V	
cs[0]	151	input	LVTTL/LVCMOS		Y
cs[1]	152	input	LVTTL/LVCMOS		Y
cs[2]	153	input	LVTTL/LVCMOS		Y
cs[3]	154	input	LVTTL/LVCMOS		Y
GND_INT	155	gnd			
cs[4]	156	input	LVTTL/LVCMOS		Y
cs[5]	157	input	LVTTL/LVCMOS		Y
d64	158	input	LVTTL/LVCMOS		Y
/denin1	159	input	LVTTL/LVCMOS		Y
VCC_INT	160	power		3.3V	
/denin	161	input	LVTTL/LVCMOS		Y
/lden	162	input	LVTTL/LVCMOS		Y
GND*	163				
GND*	164				
GND_INT	165	gnd			
ladi	166	input	LVTTL/LVCMOS		Y
/aben	167	input	LVTTL/LVCMOS		Y
laen	168	input	LVTTL/LVCMOS		Y
ga[4]	169	input	LVTTL/LVCMOS		Y
VCC_INT	170	power		3.3V	
ga[3]	171	input	LVTTL/LVCMOS		Y
ga[2]	172	input	LVTTL/LVCMOS		Y
ga[1]	173	input	LVTTL/LVCMOS		Y
ddc_even_/intrpt	174	input	LVTTL/LVCMOS		Y
ddc_odd_/intrpt	175	input	LVTTL/LVCMOS		Y
GND_INT	176	gnd			
TDI	177	input			
nCE	178	input			
DCLK	179	bidir			
DATA0	180	input			
/l_reset	181	output	LVTTL/LVCMOS		Y
fifo_even_/ef	182	input	LVTTL/LVCMOS		Y
fifo_even_/ff	183	input	LVTTL/LVCMOS		Y
r/w	184	input	LVTTL/LVCMOS		Y
fifo_even_/pae	185	input	LVTTL/LVCMOS		Y
/l_data_ready	186	output	LVTTL/LVCMOS		Y

Pin Name/ Usage C-FPGA V12	Loca-tion	Dir.	I/O Standard	Voltage	User Defined
ddc_even_/oea	187	output	LVTTL/LVCMOS		Y
l_data_ack	188	input	LVTTL/LVCMOS		Y
VCC_INT	189	power		3.3V	
fifo_even_/paf	190	input	LVTTL/LVCMOS		Y
ddc_even_/oeb	191	output	LVTTL/LVCMOS		Y
ddc_odd_/oea	192	output	LVTTL/LVCMOS		Y
ddc_odd_/oeb	193	output	LVTTL/LVCMOS		Y
ga[0]	194	input	LVTTL/LVCMOS		Y
ld[7]	195	bidir	LVTTL/LVCMOS		Y
ld[6]	196	bidir	LVTTL/LVCMOS		Y
GND_INT	197	gnd			
ld[5]	198	bidir	LVTTL/LVCMOS		Y
ld[4]	199	bidir	LVTTL/LVCMOS		Y
ld[3]	200	bidir	LVTTL/LVCMOS		Y
ld[2]	201	bidir	LVTTL/LVCMOS		Y
ld[1]	202	bidir	LVTTL/LVCMOS		Y
ld[0]	203	bidir	LVTTL/LVCMOS		Y
la[7]	204	input	LVTTL/LVCMOS		Y
VCC_INT	205	power		3.3V	
la[6]	206	input	LVTTL/LVCMOS		Y
la[5]	207	input	LVTTL/LVCMOS		Y
la[4]	208	input	LVTTL/LVCMOS		Y
/reset	209	input	LVTTL/LVCMOS		Y
fifo_full	210	input	LVTTL/LVCMOS		Y
clock40	211	input	LVTTL/LVCMOS		Y
ddc_clk	212	input	LVTTL/LVCMOS		Y
GND*	213				
la[3]	214	input	LVTTL/LVCMOS		Y

Pin Name/ Usage C-FPGA V12	Loca-tion	Dir.	I/O Standard	Voltage	User Defined
la[2]	215	input	LVTTL/LVCMOS		Y
GND_INT	216	gnd			
la[1]	217	input	LVTTL/LVCMOS		Y
la[0]	218	input	LVTTL/LVCMOS		Y
ld[15]	219	bidir	LVTTL/LVCMOS		Y
ld[14]	220	bidir	LVTTL/LVCMOS		Y
ld[13]	221	bidir	LVTTL/LVCMOS		Y
ld[12]	222	bidir	LVTTL/LVCMOS		Y
ld[11]	223	bidir	LVTTL/LVCMOS		Y
VCC_INT	224	power		3.3V	
ld[10]	225	bidir	LVTTL/LVCMOS		Y
ld[9]	226	bidir	LVTTL/LVCMOS		Y
ld[8]	227	bidir	LVTTL/LVCMOS		Y
la[15]	228	input	LVTTL/LVCMOS		Y
la[14]	229	input	LVTTL/LVCMOS		Y
la[13]	230	input	LVTTL/LVCMOS		Y
la[12]	231	input	LVTTL/LVCMOS		Y
GND_INT	232	gnd			
la[11]	233	input	LVTTL/LVCMOS		Y
la[10]	234	input	LVTTL/LVCMOS		Y
la[9]	235	input	LVTTL/LVCMOS		Y
fifo_odd_/ef	236	input	LVTTL/LVCMOS		Y
la[8]	237	input	LVTTL/LVCMOS		Y
fifo_odd_/pae	238	input	LVTTL/LVCMOS		Y
fifo_odd_/paf	239	input	LVTTL/LVCMOS		Y
fifo_odd_/ff	240	input	LVTTL/LVCMOS		Y

8.3 Pin location version L-FPGA

Pin Name/ Usage L-FPGA	Location	Dir.	I/O Standard	Voltage	User Defined
TCK	1	input			
CONF_DONE	2	bidir			
nCEO	3	output			
TDO	4	output			
VCC_IO	5	power		3.3V	
VCC_INT	6	power		3.3V	
GND*	7				
data[8]	8	input	LVTTL/LVCMOS		Y
data[10]	9	input	LVTTL/LVCMOS		Y
data[11]	10	input	LVTTL/LVCMOS		Y
GND*	11				
data[12]	12	input	LVTTL/LVCMOS		Y
data[13]	13	input	LVTTL/LVCMOS		Y
GND*	14				
GND_IO	15	gnd			
GND_INT	16	gnd			
data[14]	17	input	LVTTL/LVCMOS		Y
data[15]	18	input	LVTTL/LVCMOS		Y
GND*	19				
lack	20	input	LVTTL/LVCMOS		Y
/lackrx	21	output	LVTTL/LVCMOS		Y
lacktx	22	output	LVTTL/LVCMOS		Y
l1clk	23	output	LVTTL/LVCMOS		Y
VCC_IO	24	power		3.3V	
VCC_INT	25	power		3.3V	
data[7]	26	input	LVTTL/LVCMOS		Y
GND*	27				
l[3]	28	output	LVTTL/LVCMOS		Y
l[2]	29	output	LVTTL/LVCMOS		Y
l[1]	30	output	LVTTL/LVCMOS		Y
l[0]	31	output	LVTTL/LVCMOS		Y
/lrx	32	output	LVTTL/LVCMOS		Y
ltx	33	output	LVTTL/LVCMOS		Y
TMS	34	input			
nSTATUS	35	bidir			
data[48]	36	input	LVTTL/LVCMOS		Y
data[49]	37	input	LVTTL/LVCMOS		Y
data[50]	38	input	LVTTL/LVCMOS		Y
data[51]	39	input	LVTTL/LVCMOS		Y
GND_IO	40	gnd			
data[52]	41	input	LVTTL/LVCMOS		Y
data[53]	42	input	LVTTL/LVCMOS		Y
data[54]	43	input	LVTTL/LVCMOS		Y
data[55]	44	input	LVTTL/LVCMOS		Y
VCC_IO	45	power		3.3V	
data[56]	46	input	LVTTL/LVCMOS		Y

Pin Name/ Usage L-FPGA	Location	Dir.	I/O Standard	Voltage	User Defined
data[57]	47	input	LVTTL/LVCMOS		Y
data[58]	48	input	LVTTL/LVCMOS		Y
data[59]	49	input	LVTTL/LVCMOS		Y
GND_IO	50	gnd			
data[60]	51	input	LVTTL/LVCMOS		Y
VCC_INT	52	power		3.3V	
VCC_INT	53	power		3.3V	
sampleclk	54	input	LVTTL/LVCMOS		Y
clk80	55	input	LVTTL/LVCMOS		Y
GND+	56				
GND_INT	57	gnd			
GND_INT	58	gnd			
data[61]	59	input	LVTTL/LVCMOS		Y
data[62]	60	input	LVTTL/LVCMOS		Y
VCC_IO	61	power		3.3V	
data[63]	62	input	LVTTL/LVCMOS		Y
data[32]	63	input	LVTTL/LVCMOS		Y
data[33]	64	input	LVTTL/LVCMOS		Y
data[34]	65	input	LVTTL/LVCMOS		Y
GND_IO	66	gnd			
data[35]	67	input	LVTTL/LVCMOS		Y
data[36]	68	input	LVTTL/LVCMOS		Y
data[37]	69	input	LVTTL/LVCMOS		Y
data[38]	70	input	LVTTL/LVCMOS		Y
VCC_IO	71	power		3.3V	
data[39]	72	input	LVTTL/LVCMOS		Y
data[40]	73	input	LVTTL/LVCMOS		Y
nCONFIG	74	input			
VCC_INT	75	power		3.3V	
MSEL1	76	input			
MSEL0	77	input			
data[41]	78	input	LVTTL/LVCMOS		Y
data[42]	79	input	LVTTL/LVCMOS		Y
data[43]	80	input	LVTTL/LVCMOS		Y
data[44]	81	input	LVTTL/LVCMOS		Y
data[45]	82	input	LVTTL/LVCMOS		Y
data[46]	83	input	LVTTL/LVCMOS		Y
GND_INT	84	gnd			
GND_IO	85	gnd			
data[47]	86	input	LVTTL/LVCMOS		Y
GND*	87				
GND*	88				
GND*	89				
data[16]	90	input	LVTTL/LVCMOS		Y
data[17]	91	input	LVTTL/LVCMOS		Y
data[18]	92	input	LVTTL/LVCMOS		Y

Pin Name/ Usage L-FPGA	Loca-tion	Dir.	I/O Standard	Voltage	User Defined
VCC_INT	93	power		3.3V	
VCC_IO	94	power		3.3V	
data[19]	95	input	LVTTL/LVCMOS		Y
data[20]	96	input	LVTTL/LVCMOS		Y
data[21]	97	input	LVTTL/LVCMOS		Y
data[22]	98	input	LVTTL/LVCMOS		Y
data[23]	99	input	LVTTL/LVCMOS		Y
data[24]	100	input	LVTTL/LVCMOS		Y
data[25]	101	input	LVTTL/LVCMOS		Y
data[26]	102	input	LVTTL/LVCMOS		Y
GND_INT	103	gnd			
GND_IO	104	gnd			
TDI	105	input			
nCE	106	input			
DCLK	107	bidir			
DATA0	108	input			
GND*	109				
data[9]	110	input	LVTTL/LVCMOS		Y
l_data_ack	111	output	LVTTL/LVCMOS		Y
fifo_full	112	output	LVTTL/LVCMOS		Y
l_data_ready	113	input	LVTTL/LVCMOS		Y
fifo_empty	114	output	LVTTL/LVCMOS		Y
VCC_IO	115	power		3.3V	
fifo_half_full	116	output	LVTTL/LVCMOS		Y
data[27]	117	input	LVTTL/LVCMOS		Y
data[28]	118	input	LVTTL/LVCMOS		Y
data[29]	119	input	LVTTL/LVCMOS		Y

Pin Name/ Usage L-FPGA	Loca-tion	Dir.	I/O Standard	Voltage	User Defined
data[30]	120	input	LVTTL/LVCMOS		Y
data[31]	121	input	LVTTL/LVCMOS		Y
/reset	122	input	LVTTL/LVCMOS		Y
VCC_INT	123	power		3.3V	
GND+	124				
clk40	125	input	LVTTL/LVCMOS		Y
GND+	126				
GND_INT	127	gnd			
GND*	128				
GND_IO	129	gnd			
data[0]	130	input	LVTTL/LVCMOS		Y
data[1]	131	input	LVTTL/LVCMOS		Y
data[2]	132	input	LVTTL/LVCMOS		Y
data[3]	133	input	LVTTL/LVCMOS		Y
VCC_IO	134	power		3.3V	
data[4]	135	input	LVTTL/LVCMOS		Y
data[5]	136	input	LVTTL/LVCMOS		Y
data[6]	137	input	LVTTL/LVCMOS		Y
GND*	138				
GND_IO	139	gnd			
GND*	140				
GND*	141				
GND*	142				
GND*	143				
GND*	144				

9. Test Engineering

9.1 File based test benches

The test benches used in that project are written in VHDL and provide the following simulation setup:

- Generate periodic clock signals for driving simulation and clocked circuit models.
- Obtain stimuli vectors and apply them to the model under test (MUT) at well-defined moments of time.
- Acquire the signal waveform which emerge from the MUT as actual response vectors.
- Obtain expected response vectors and use as the reference against which the accrual responses are compared.
- Establish a simulation report that points to differences.

Nr.	Purpose of file based test benches	Ref.
12.1	Test benches are developed for verification: ■ Functional test of all data transmission modes ■ Interface tests of DDC, FIFO, and VME: Enabling, acquisition modes, FIFO forwarding modes, DDC formats ■ Configuration data distribution: Static DDC registers, SOF, COF ■ IRQ operation via VME bus ■ Board addressing ■ Monitor information	[-]

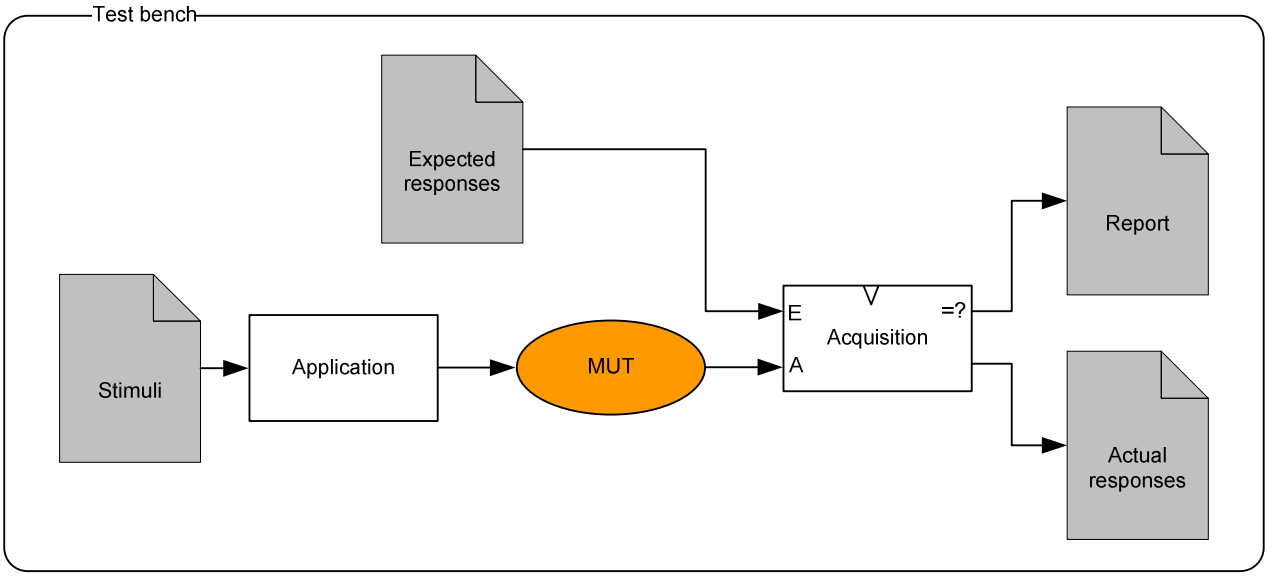


Figure 20: File based test bench with stimuli application, model under test (MUT), response acquisition, comparison of actual and expected responses, and reporting.

Nr.	Structure of file based test benches	Ref.
12.2	The main building blocks of the test bench are ■ Declaration of MUT: C-FPGA and/or L-FPGA ■ Parameter settings: Constants, timing definitions, record declaration ■ Processing functions: • GetStimuliRecord, GetExpectedResponseRecord, PutSimulationReportTrace, PutSimulationReportFailure • GetIFStimuliRecord, GetExpectedResponseRecord, PutIFSimulationsReportTrace, PutIFSimulationReportFailure • GetLinkExpectedResponseRecord, PutLinkSimulationsReportTrace, PutLinkSimulationReportFailure ■ Instantiation of MUT ■ Clock processes (40 MHz, 80 MHz, 30 MHz, …) ■ File handler: open and close of files depending on the test bench status ■ Stimuli application process ■ Actual response acquisition and reporting process	[-]
12.3	The vectors (stimuli, expected and actual responses) are all stored as files with tables. Every column is associated with an input pin and every line describes one time stamp. ■ The stimuli vectors used are separated as file for all VME signals and one for the remaining interface signals. ■ The expected vectors are divided as responses of VME bus, DDC/FIFO/Selector interface, and SHARC link signals. ■ The structure of the actual responses are comparable with those of the expected responses.	[-]

Nr.	Test bench variations	Ref.
12.4	Variations of the test bench have been used for setups with ■ Existing implementation (Main and SHARC-Link FPGA) for generation of expected responses ■ Behaviour implementation of C-FPGA V11 and L-FPGA ■ Behaviour implementation of C-FPGA V12 and L-FPGA ■ Gate-level implementation of C-FPGA V11 and behaviour L-FPGA ■ Gate-level implementation of C-FPGA V12 and behaviour L-FPGA ■ Behaviour implementation of C-FPGA V11 and gate-level implementation L-FPGA ■ Mixed version with behaviour implementation of C-FPGA V11 with existing SHARC-Link FPGA ■ Mixed version with existing implementation of Main FPGA with behaviour implementation of L-FPGA	[-]

9.2 Hardware tests

Nr.	Hardware tests	Ref.
13.1	Tests on the target board ■ Hardware tests are based on the PSI test infrastructure. ■ The tests verify the basic functionality of the real hardware. Its operation and data forwarding modes are in accordance with the existing ramp and configuration tests software.	[16]
13.2	Specific HW test protocol: ■ Systems boot with board identification via VME bus ■ Register write and read via VME bus using the entire space of the register map ■ Write FIFO entry and read back data ■ Write complete FIFO and read the entire FIFO content ■ Acquire DDC data in FIFO and read back data via VME ■ Acquire DDC data and forward it over SHARC link ■ Configure DDC via C-Bus interface and observe scaled data stored in the FIFO ■ Test different acquisition modes with as triggered and closed-orbit configuration, or as the four gate modes [8.2] and forwarding modes [8.3] ■ Tests with both board version: V11 and V12 ■ Supplementary tests	[-]

10. Design Files

10.1 Source files

Nr.	File name	Function
14.1	../FPGA/sourcecode/cFPGA.vhd	C-FPGA driver interfaces
14.2	../FPGA/sourcecode/cFPGAtop.vhd	C-FPGA functional top level unit
14.3	../FPGA/sourcecode/CentralControl.vhd	Central control unit
14.4	../FPGA/sourcecode/VMEinterface.vhd	VME Interface unit
14.5	../FPGA/sourcecode/SHARCinterface.vhd	SHARC Interface unit
14.6	../FPGA/sourcecode/DDCFIFOSelectroInterface.vhd	Interface unit to DDC, FIFO and selectors
14.7	../FPGA/sourcecode/GateSynchronization.vhd	Gate Synchronization unit
14.8	../FPGA/sourcecode/boardAddress.vhd	Board Address unit
14.9	../FPGA/sourcecode/lFPGA.vhd	L-FPGA driver interfaces
14.10	../FPGA/sourcecode/lFPGAtop.vhd	L-FPGA functional top level unit
14.11	../FPGA/sourcecode/Const.vhd	Package with constants and defines

10.2 Test benches

Nr.	File name	Function
15.1	../FPGA/sourcecode/System_tb.vhd	Basic test bench
15.2	../FPGA/sourcecode/systemMainV11_tb.vhd	Test bench for existing version V11 as Golden Model
15.3	../FPGA/sourcecode/systemMainV12_tb.vhd	Test bench for existing version V12 as Golden Model
15.4	../FPGA/sourcecode/systemV11_tb.vhd	Test bench for implementation V11 verification
15.5	../FPGA/sourcecode/systemV12_tb.vhd	Test bench for implementation V12 verification

10.3 Modelsim scripts

Nr.	File name	Function
16.1	../simvectores/stimuli.asc	Stimuli file of the VME signals
16.2	../simvectores/expresp.asc	Expected responses of the VME signals
16.3	../simvectores/report.asc	Report generated by the test bench on the VME signal comparison
16.4	../simvectores/IFstimuli.asc	Stimuli file of the DDC/FIFO/Selector interface signals
16.5	../simvectores/IFexpresp.asc	Expected responses of the DDC/FIFO/Selector interface signals
16.6	../simvectores/IFreport.asc	Report generated by the test bench on the DDC/FIFO/Selector interface signal comparison
16.7	../simvectores/LINKexpresp	Expected responses of the SHARC link signals
16.8	../simvectores/LINKreport	Report generated by the test bench on the SHARC signal comparison

For gate-level simulation a specific copy of a simvectores directory is stored in the dedicated Quartus project directory.

10.4 Modelsim scripts

Nr.	File name	Function
17.1	../FPGA/scripts/compileExisting.scr	Reference design compile scripts
17.2	../FPGA/scripts/compileBehavirol.scr	Behaviour compilation for simulation
17.3	../FPGA/scripts/compileMixed.scr	Mixed behaviour and reference design compilation for verification
17.4	../FPGA/scripts/compilePostSynthesis.scr	Post synthesis compilation for gate level verification

10.5 Backend Quartus synthesis project files

Nr.	File name	Function
18.1	../FPGA/QDRcfpgaV11/cFPGAv11.qpf	C-FPGA V11 Quartus project file
18.1	../FPGA/QDRcfpgaV11/cFPGAv11.pin	C-FPGA V11 pin file
18.2	../FPGA/QDRcfpgaV12/cFPGAv12.qpf	C-FPGA V12 Quartus project file
18.2	../FPGA/QDRcfpgaV12/cFPGAv12.pin	C-FPGA V12 pin file
18.3	../FPGA/lfpga/lFPGA.qpf	L-FPGA Quartus project file
18.3	../FPGA/lfpga/lFPGA.pin	L-FPGA pin file

11. Altera Synthesis using Quartus II Version 7.2

11.1 Synthesis results

Nr.	Synthesis	LE	Timing	IO
19.1	C-FPGA V11	1176 (=68%)	51.8 MHz	177
	V12	1229 (=71%)	53.1 MHz	185
19.2	L-FPGA	137 (=24%)	> 90 MHz	83

11.2 Synthesis reports C-FPGA V11

Nr.	File name	Function
20.1	../FPGA/QDRcfpgaV11/cFPGAv11.map.rpt	Analysis & Synthesis report for cFPGAv11
20.2	../FPGA/QDRcfpgaV11/cFPGAv11.fit.rpt	Fitter report for cFPGAv11
20.3	../FPGA/QDRcfpgaV11/cFPGAv11.asm.rpt	Assembler report for cFPGAv11
20.4	../FPGA/QDRcfpgaV11/cFPGAv11.tan.rpt	Classic Timing Analyzer report for cFPGAv11
20.5	../FPGA/QDRcfpgaV11/cFPGAv11.flow.rpt	Flow report for cFPGAv11
20.6	../FPGA/QDRcfpgaV11/cFPGAv11.eda.rpt	EDA Netlist Writer report for cFPGAv11
20.7	../FPGA/QDRcfpgaV11/cFPGAv11.eda.rpt	Nativelink Simulation process report for cFPGAv11

11.3 Synthesis reports C-FPGA V12

Nr.	File name	Function
21.1	../FPGA/QDRcfpgaV12/cFPGAv12.map.rpt	Analysis & Synthesis report for cFPGAv12
21.2	../FPGA/QDRcfpgaV12/cFPGAv12.fit.rpt	Fitter report for cFPGAv12
21.3	../FPGA/QDRcfpgaV12/cFPGAv12.asm.rpt	Assembler report for cFPGAv12
21.4	../FPGA/QDRcfpgaV12/cFPGAv12.tan.rpt	Classic Timing Analyzer report for cFPGAv12
21.5	../FPGA/QDRcfpgaV12/cFPGAv12.flow.rpt	Flow report for cFPGAv12
21.6	../FPGA/QDRcfpgaV12/cFPGAv12.eda.rpt	EDA Netlist Writer report for cFPGAv12
21.7	../FPGA/QDRcfpgaV12/cFPGAv12.eda.rpt	Nativelink Simulation process report for cFPGAv12

11.4 Synthesis reports L-FPGA

Nr.	File name	Function
22.1	../FPGA/lfpga/lfpga.map.rpt	Analysis & Synthesis report for lfpga
22.2	../FPGA/lfpga/lfpga.fit.rpt	Fitter report for lfpga
22.3	../FPGA/lfpga/lfpga.asm.rpt	Assembler report for lfpga
22.4	../FPGA/lfpga/lfpga.tan.rpt	Classic Timing Analyzer report for lfpga
22.5	../FPGA/lfpga/lfpga.flow.rpt	Flow report for lfpga
22.6	../FPGA/lfpga/lfpga.eda.rpt	EDA Netlist Writer report for lfpga
22.7	../FPGA/lfpga/lfpga.eda.rpt	Nativelink Simulation process report for lfpga