RPW Instrument

LFR Low Frequency Receiver
FPGA Architecture Design

Prepared by:	Function:	Signature:	Date
J-C. PELLION	VHDL developer		
A. Jeandet	Lead Engineer VHDL developer		
Approved by:	**Function:**	**Signature:**	**Date**
P. LEROY	Project Manager Flight Software developer		
For application:	**Function:**	**Signature:**	**Date**

Ecole Polytechnique
Route de Saclay
91128 Palaiseau CEDEX

Change Record

Ver.	Date	Authors	Modifications
1.7	05 DEC 2014	JCPE	Update LFRs Register
1.6		JCPE	Update LFRs Register
1.5		PLE	Filter section updated
1.4	09 MAY 2014	PLE	
1.3	07 MAY 2013	JCPE	Update to match Dev Plan Requirement
1.2	05 DEC 2013	JCPE	Update
1.1	23 SEP 2013	JCPE	Include update from Paul LEROY
1.0	04 SEP 2013	JCPE	First Version

Acronym List

Acronym	Definition	Acronym	Definition
FPGA	Field Programmable Gate Array		
LFR	Low Frequency Receiver		
MEB	Main Electronics Box		
N/A	Not Applicable		
RPW	Radio and Plasma Waves (experiment)		
SPW	SpaceWire		
TBC	To Be Confirmed		
TBD	To Be Defined		
TBW	To Be Written		

Table of Contents

List of figures

1 GENERAL

1.1 Scope of the Document

The purpose of this document is to present the architecture and design requirements of the LFRôs FPGA.

1.2 Applicable Documents

This document responds to the requirements of the documents listed in the following table:

#	Reference	Title of the document	Rev

1.3 Reference Documents

This document is based on the documents listed in the following table:

#	Reference	Title of the document	Date / Rev
RD1	RPW-MEB-LFR-00003-1-Rev6	LFR Specification	1.7
RD2	Rapport de Stage_Alexis	Système d'acquisition numérique pour analyseur de données embarqué	30 JUN 2009
RD3		Software System Specification	
RD4		Software Requirements Specification	

1.4 Responsibilities

Name	Function	Institute	Phone number	Email
J-C. Pellion	VHDL developper	LPP	+33 1 69 33 -- --	jean-christophe.pellion @lpp.polytechnique.fr
P. Leroy	Project Manager			paul.leroy@lpp.polytechnique.fr

2 LFR FPGA

2.1 ARCHITECTURE DEFINITION

2.1.1 Functional view

The operations realized by LFR are defined into the RD1 documents. Figure 1 shows the functional block view of LFR FPGA. Data acquisition is done by 8 external ADCs.

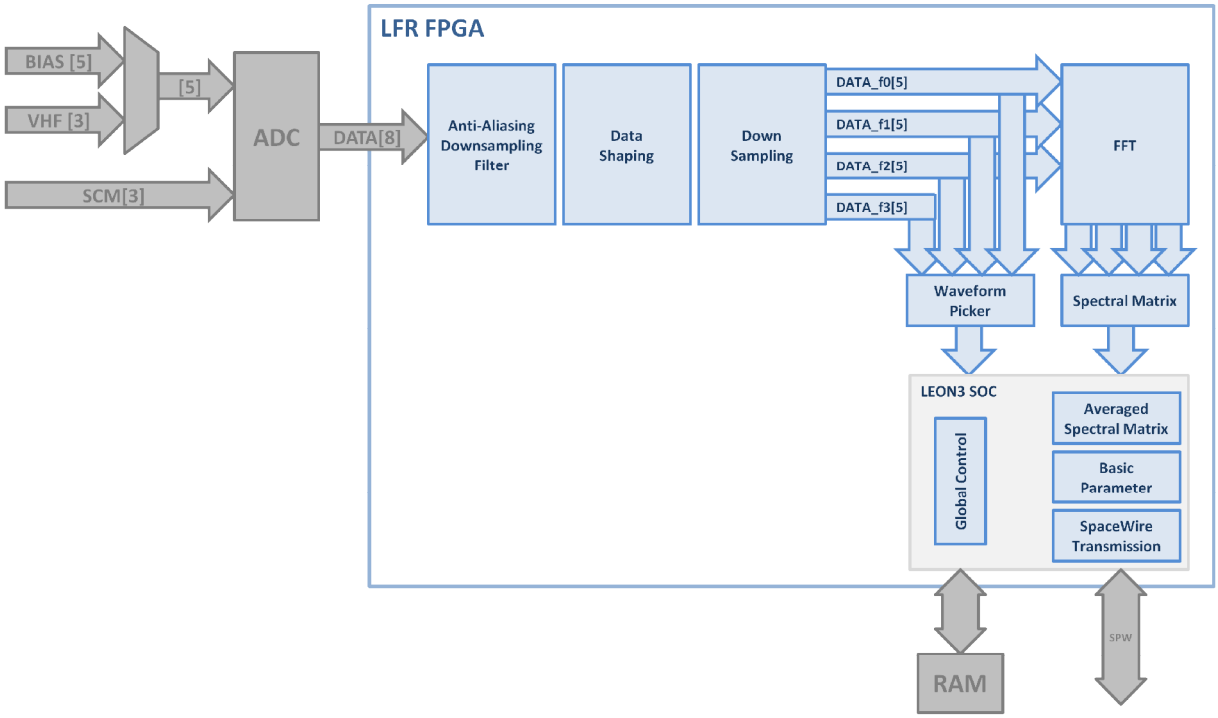


Figure 1: LFR FPGA functional view

In the FPGA, the data are filtered, shaped and downsampled. After these preliminary steps, the data are sent to the external RAM by the WaveForm Picker module. For each channel of frequencies, FFT and Spectral Matrices are computed (in hardware) and sent to the Leon3 SoC via DMA transfers.

By Software (Leon3), Spectral Matrices are averaged and Basic Parameters are extracted [RD1]. The flight software controls all hardware modules and interacts with the DPU through the fully redundant SpaceWire link.

2.1.2 FPGA interfaces

2.1.2.1 External Devices

2.1.2.1.1 RAM

The RAM used on the LFR EM board is the following: Cypress CYC1061DV33. It is a static ram 1M*16bit. The documentation can be downloaded from the official website. Figure 2 shows the RAM block diagram. The flight RAM will be the Aeroflex UT8ER1M32.

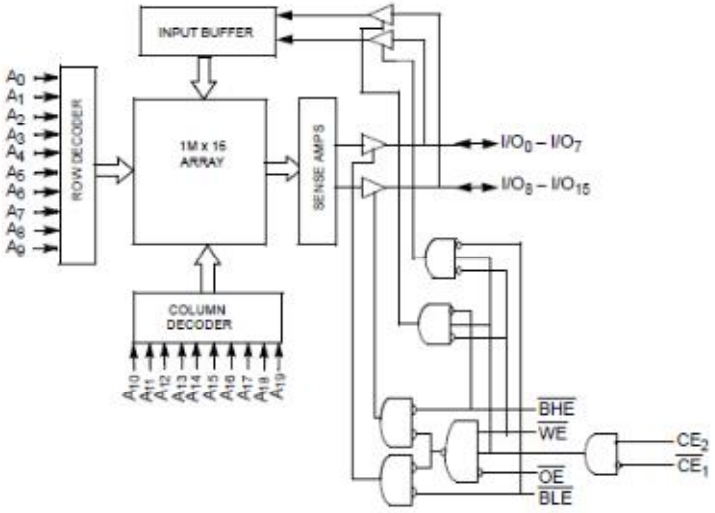


Figure 2: RAM's block diagram

The UT8ER1M32 memory controller is a modified Gaislerôs IP SRCTRL provided by TDS team. The modifications allow the memory controller to:

- Perform memory wash on startup (needed for correct EDAC operation)
- Not modify the RAM input signal during scrubbing cycles
- Startup EDAC control register write
- Runtime EDAC control register read/write

This IP hasnôt been tested yet by the LPP team. A dedicated board has been designed to validate the memory and its interfaces, more details will be given on the next revision of this document.

2.1.2.1.2 ADC

The ADCs used on the EM board are proto version of the RHF1401 from ST-microelectronics. RHF1401 is a 14 bits analog to digital converter. Figure 3 shows the block diagram and Figure 4 shows the protocol to obtained the Data.

In our case, 8 ADCs are used in parallel. The Data pins and clock are commons. The control is realized by the OEB vector. Only one OEB must be activated at a time.

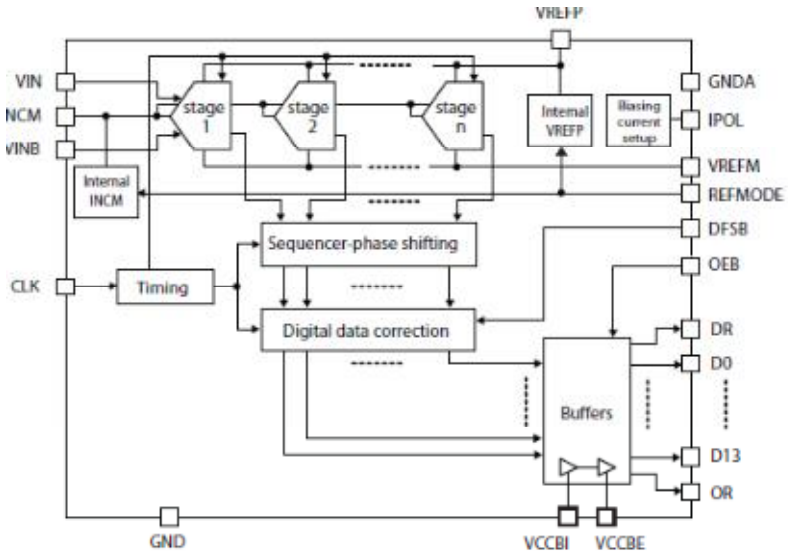


Figure 3: ADC's block diagram

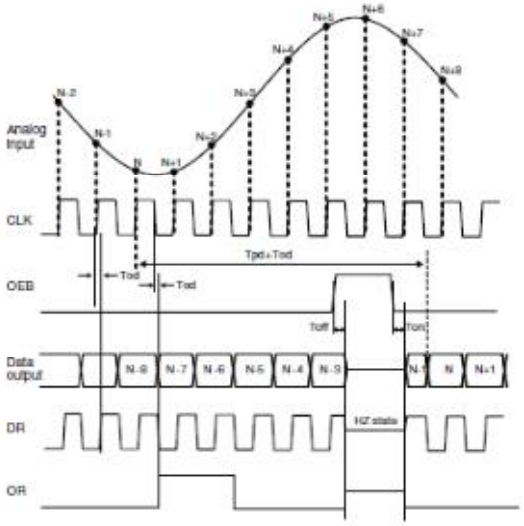


Figure 4: ADC's protocol

2.1.2.2 Pinout and Package

2.1.2.2.1 Engineering Model

The FPGA used for the Engineering model is the following:

Family type: ProASIC3E

Die: A3PE3000

Package: 324 FBGA

The Table below defines the Pinout of the EM FPGA.

NAME	Pin	Direction	Description
General Purpose			
clk_49_152MHz	D5	IN	Clock input at 49.152MHz
clk100MHz	B3	IN	Clock input at 100MHz
reset	N18	IN	Reset signal active at low value

RAM			
Address		OUT	20 bits of RAM address
[0]	H16		
[1]	J15		
[2]	B18		
[3]	C17		
[4]	C18		
[5]	U2		
[6]	U3		
[7]	R5		
[8]	N11		
[9]	R13		
[10]	V13		
[11]	U13		
[12]	V15		
[13]	V16		
[14]	V17		
[15]	N1		
[16]	R3		
[17]	P4		
[18]	N3		
[19]	M7		
Data		IN/OUT	32 bits of RAM data
[0]	P17		
[1]	R18		
[2]	T18		
[3]	J13		
[4]	T13		
[5]	T12		
[6]	R12		
[7]	T11		
[8]	N2		
[9]	P1		
[10]	R1		
[11]	T1		
[12]	M4		
[13]	K1		
[14]	J1		
[15]	H1		
[16]	H15		
[17]	G15		
[18]	H13		
[19]	G12		
[20]	V14		
[21]	N9		
[22]	M13		
[23]	M15		
[24]	J17		
[25]	K15		
[26]	J14		
[27]	U18		
[28]	H18		
[29]	J18		

[30] [31]	G17 F18		
nSRAM_BE0	U12	OUT	RAM Bank0 Enable (low active)
nSRAM_BE1	K18	OUT	RAM Bank0 Enable (low active)
nSRAM_BE2	K12	OUT	RAM Bank0 Enable (low active)
nSRAM_BE3	F17	OUT	RAM Bank0 Enable (low active)
nSRAM_CE	M6	OUT	RAM Chip Enbale (low active)
nSRAM_OE	N12	OUT	RAM Ouput Enbale (low active)
Space Wire			
Spw1_din	D6	IN	Data In of Space Wire link1
Spw1_sin	C6	IN	Strobe In of Space Wire link1
Spw1_dout	C16	OUT	Data Out of Space Wire link1
Spw1_sout	C4	OUT	Strobe Out of Space Wire link1
Spw2_din	E6	IN	Data In of Space Wire link2
Spw2_sin	C15	IN	Strobe In of Space Wire link2
Spw2_dout	B7	OUT	Data Out of Space Wire link2
Spw2_sout	D7	OUT	Strobe Out of Space Wire link2
ADC			
bias_fail_sw	A3	OUT	Bias/Fail switch control
ADC_OEB_bar_CH [1] [2] [3] [4] [5] [6] [7]	 A14 A10 B10 B13 D13 A11 B12	OUT	Output Enable. One for each ADC (active low)
ADC_smpclk	A15	OUT	Sample Clock for data acquisition
ADC_data [0] [1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13]	 A16 B16 A17 C12 B17 C13 D15 E15 D16 F16 F15 G16 F13 G13	IN	
UARTs			
TAG1	J12	IN	AHB UART Rx
TAG3	L16	OUT	AHB UART Tx
TAG2	K13	IN	APB UART Rx
TAG4	L15	OUT	APB UART Rx
Others			
TAG5	M16	IN/OUT	Unused

TAG6	L13	IN/OUT	Unused
TAG7	P6	IN/OUT	Unused
TAG8	R6	IN/OUT	Unused
TAG9	T4	IN/OUT	Unused
LED [0] [1] [2]	 K17 L18 M17	OUT	LEDs control

2.2 ARCHITECTURAL DESIGN

The FPGA System is composed of 4 sub-systems:

- LPP_LFR_FILTER which reads the data from ADC, filters and down sample, and finaly output 4 channel of DATA at different frequencies;
- LPP_LFR_WAVEFORM which write a part of DATA output by LPP_LFR_FILTER into the RAM in function of the configuration;
- LPP_LFR_MS which computed for each channel the Spectral Matrix and writes it into the RAM
- LEON3_SoC which controls the 3 others sub-system, interacted with the DPU through the Space Wire and computes the Averaged Matrix and Basic Parameters.

Figure 5 shows the top-level view of the LFR FPGA.

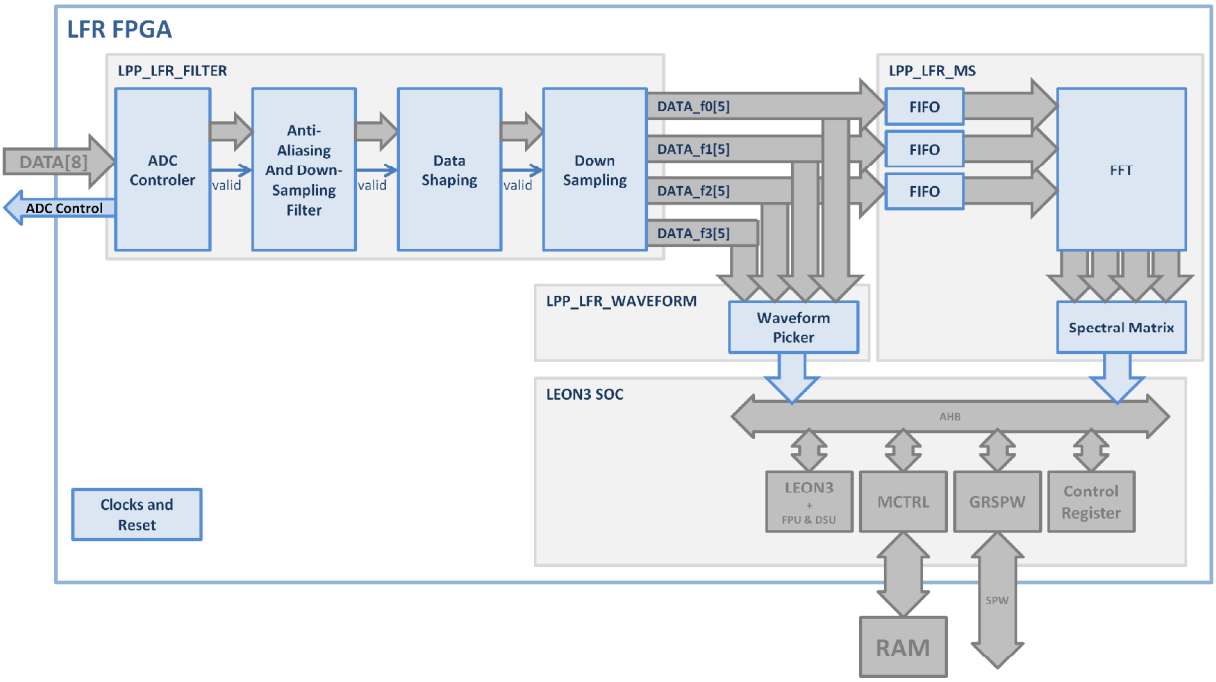


Figure 5 LFR FPGA block diagram

2.3 DETAILED DESIGN

2.3.1 Data Base

The LFR project is partly based on the Aeroflex/Gaisler IP cores library, the GRLIB. The version used in LFR is grlib-ft-fpga-grlfpu-spw-1.2.4-b4126.

In parallel of this Library, we have developed our own library of IP cores, named the VHD_Lib. This library contains all our RTL files and scripts.

The sources of the VHD_Lib are managed with Mercurial, a distributed source control management tool. LPP also uses RhodeCode, a web-based tool for repository management. All LPP VHDL sources can be consulted via the LPP server at the following address:

https://hephaistos.lpp.polytechnique.fr/rhodecode/HG_REPOSITORIES/LPP/INSTRUMENTATION/USERS/PELLION/VHD_Lib

The current folder organization of the VHD_Lib is:

- **boards** (contains the pinout and clock constraint for each board)
  - **em-LeonLPP-A3PE3kL-v3-core1** is the folder for the LFR-EM board
  - **MINI-LFR** is the folder for the MINI-LFR board
- **lib/lpp** contains all LPP IPs (lpp_waveform, lpp_matrix, etc)
- **designs** contains the different project used for simulation or synthesis
  - **LFR-em-WFP_MS** is the design for LFR EM board with the Waveform Picker and Spectral Matrix modules instantiated
  - **MINI-LFR_WFP_MS** is the same design for the MINI-LFR board
  - **LFR_simu** contains a top of LFR with a testbench to permit a global simulation of the Waveform Picker or Spectral Matrix modules
- **scripts** contains all LPP scripts used for patched the GRLIB and worked with VHD_Lib

2.3.2 Clock and Reset

Figure 6 shows the 3 clock domains of the LFR FPGA. The ADC controller module generates the acquisition signal in the 49.152 MHz clock domain. ADC data are received in the 25 MHz clock domain. All data processing is done in the 25 MHz clock domain. The LeonSoC also uses this clock.

SpaceWire transmissions are done with a 10 MHz clock. This clock is generated by dividing the initial 50 MHz master clock. The SpaceWire IP, provided by Aeroflex/Gaisler, makes this division and synchronizes all signals between the 25MHz and 10MHz domains.

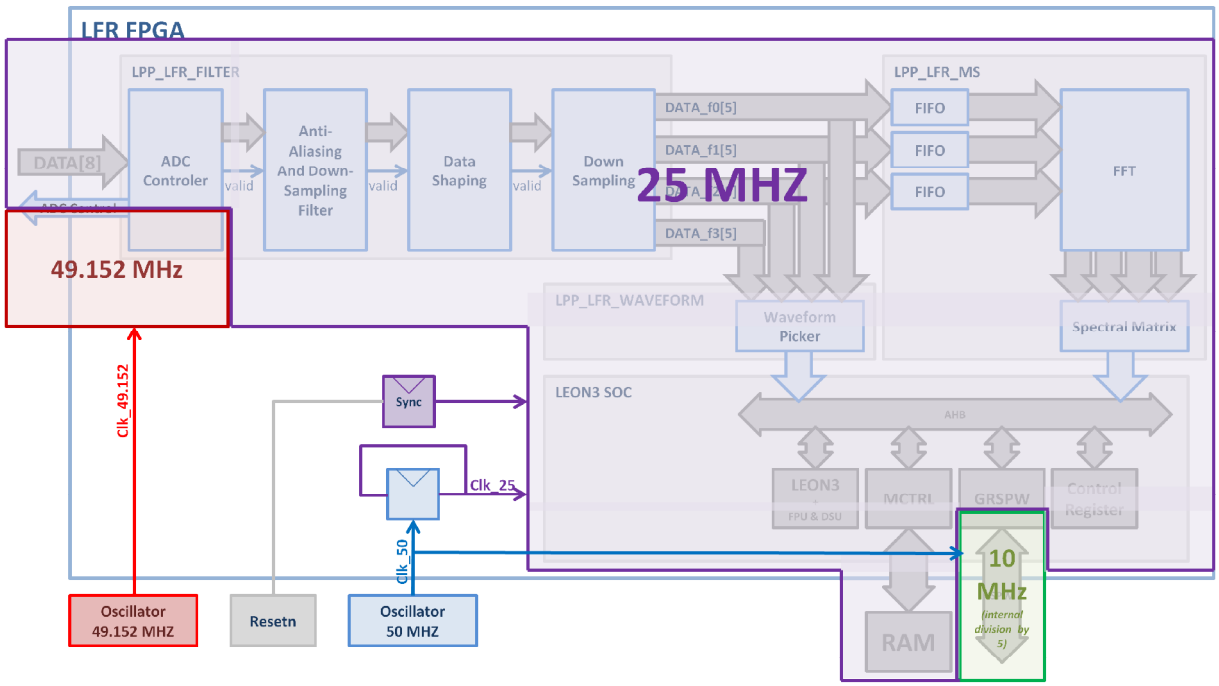


Figure 6 : LFR's Clock Domains

The 25MHz clock is obtained by dividing the input clock 50MHz.

2.3.3 Time

2.3.3.1 Time Management

The Time Management module is responsible for the local time generation of the LFR instrument. This parameter is of crucial importance as it is used to date all scientific data. The LFR local time is represented by 2 values: COARSE_TIME and FINE_TIME. FINE_TIME is a 16b signal and COARSE_TIME is a 32b signal. The FINE_TIME is incremented each period of 2^{-16} second from 0 to 65535 (1 second), and the COARSE_TIME is incremented each second. The MSB of COARSE_TIME is a special bit dedicated to time synchronization (ó0ô indicates that LFR is synchronized with the system, in accordance with the Software System Specification).

The current time can be updated by the flight software through the input register. The nominal behaviour is that each second a new ñtickò (generated upon the reception of a valid SpaceWire timecode) is sent to update the COARSE_TIME counter and reset the FINE_TIME counter. With this tick, a new COARSE_TIME value can be sent (call CTU = Coarse Time Update). If a new CTU is set, the current time counter is updated only when the next ñtickò occurs.

When the module is instantiated, the ñtickò signal is connected to the SpaceWire tick signal which is generated by the SpaceWire IP core each time a valid SpaceWire timecode has been received by the LFR board.

2.3.3.1.1 Block View

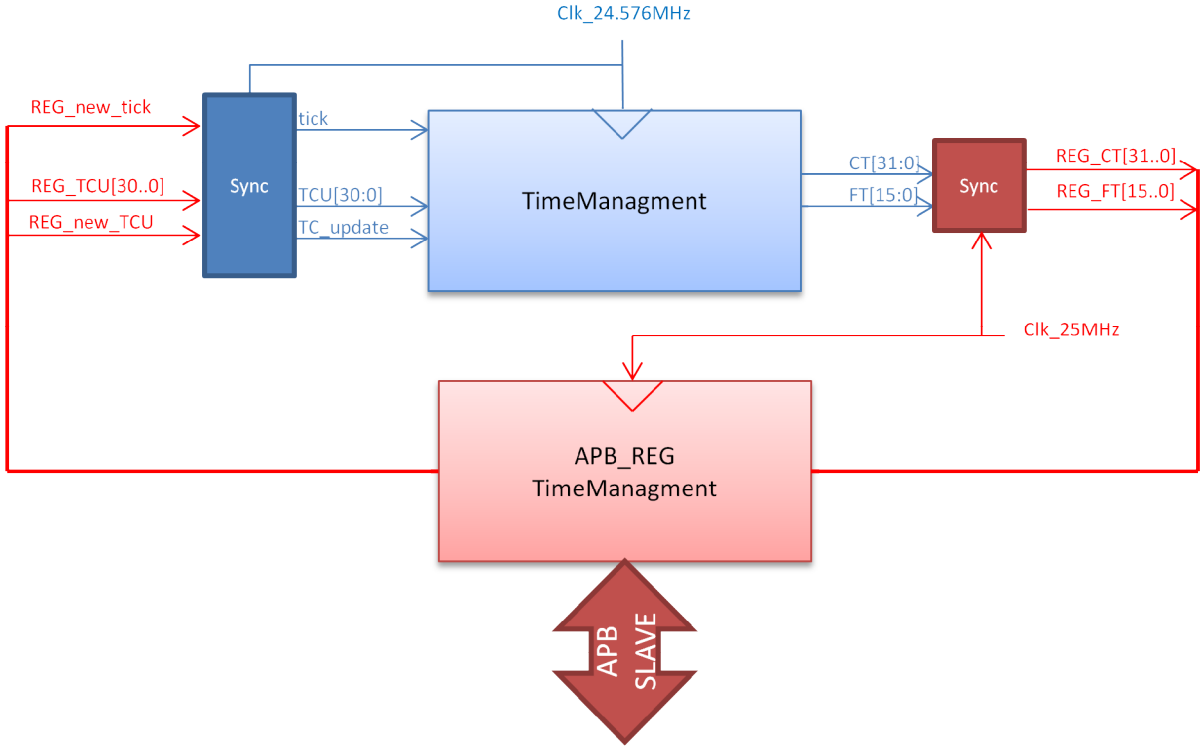


Figure 7: Time management block view

The control and observation registers are clocked on the clk_25Mhz and the coarse and fine time counters are clocked on clk_24.576MHz. There is a synchronization stage to transmit the signal from one clock domain to another.

The time management is controlled by a FSM (see Figure 8). This FSM has 3 states:

- DESYNC : a tick has not been received since more than 1.001s.
- SYNC : a tick has been received since less than 1s.
- TRANSISTION: a tick has not been received since more than 1s and less than 1.001s.

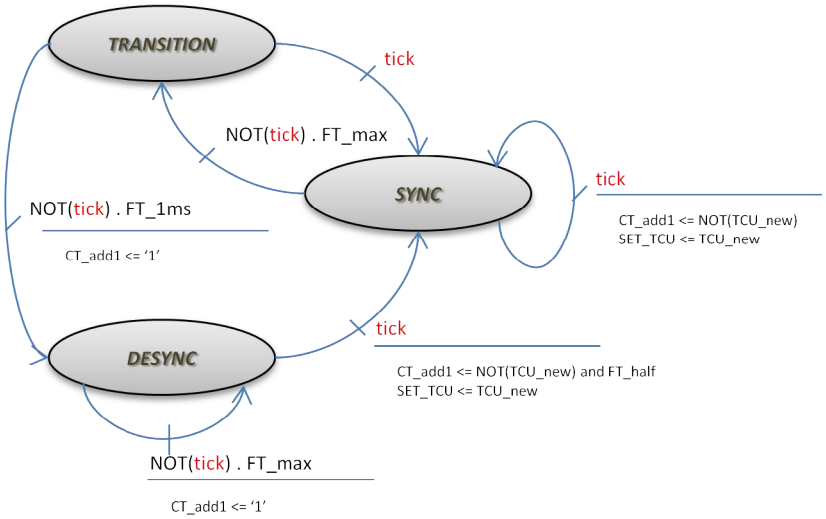


Figure 8: FSM controlling the Time Management

This FSM controls the two counters: FINE_TIME (see Figure 9) and COARSE_TIME (see Figure 10).

The FINE_TIME counter is composed of 2 counters. The first one is a counter from 0 to 374. Each clock cycle this counter is incremented, and if a tick occurs the value is reset to 0. If the value is equal to 374, the new_FT signal is set. The new_FT is set only one cycle. In conclusion, this counter divides the 24.576 Mhz by 375 to obtain a signal with a frequency of 65536Hz (=2^{16}Hz).

The second counter counts the new_FT, and is reset to 0 upon the reception of a tick. If the FSM state is SYNC or DESYNC, the FINE_TIME value is the current value of this second counter. And if the FSM state is TRANSITION, the FINE_TIME is forced to 0xFFFF (max Value).

3 signals are also generated using the FINE_TIME counter:

- FT_Max is asserted when the second counter is equal the ñmaxò value and a new_FT occurs.
- FT_half is asserted when it is more than the ñhalfò value.
- FT_1ms is asserted when it is equal to 0x0040 (i.e. the time since last RAZ is almost 1ms).

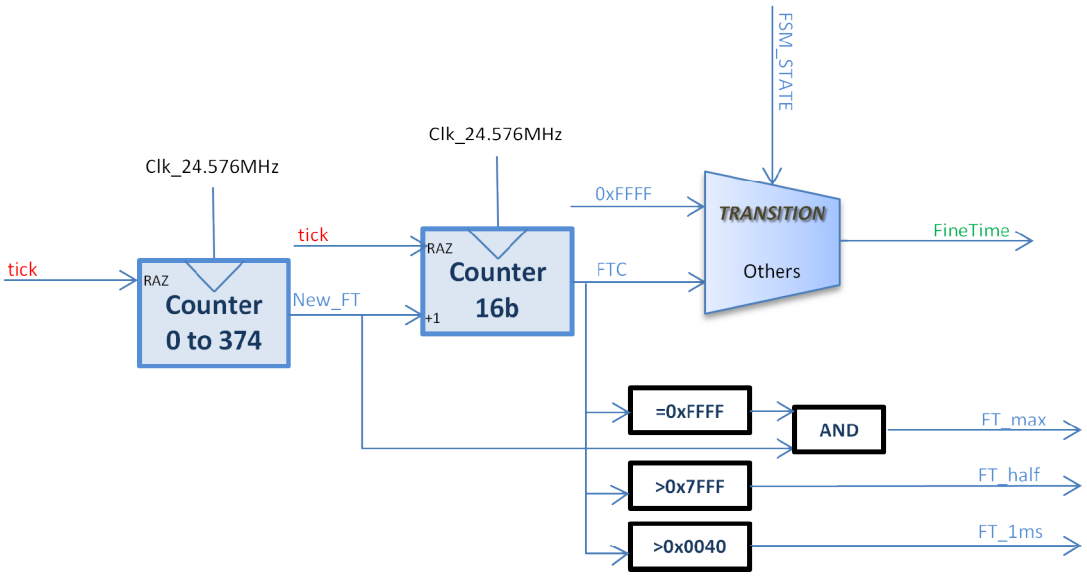


Figure 9: FINE TIME COUNTER

The COARSE_TIME counter is composed of 2 counters. The first counts seconds. Each time the FSM sends CT_add1, the current counter is updated. If the set_TCU occurs, the counter current value is set to TCU. The set_TCU signal is asserted when a new TCU is set in registers and a tick is set.

The second counter counts the time elapsed since the last synchronization, in seconds. If this value is equal to 60, the most significant bit of the COARSE_TIME is set to ó1ô After a reset, the value is 60, and the COARSE_TIME MSB is set to ó1ô This counter is a non-cyclic counter. So when the value is 60, the counter is blocked. To set the counter value to zero, a tick must occur with a new_CTU. If the counter is not equal to the full value (60), and a tick occurs, the counter is set to zero.

In conclusion, after the reset, LFR is desynchronized (COARSE_TIME bit 31 set to ó1ô). To resynchronize LFR, a tick and a TCU must occur. Once LFR is synchronized, to return in the desynchronized state, it should not have tick during more than 60s.

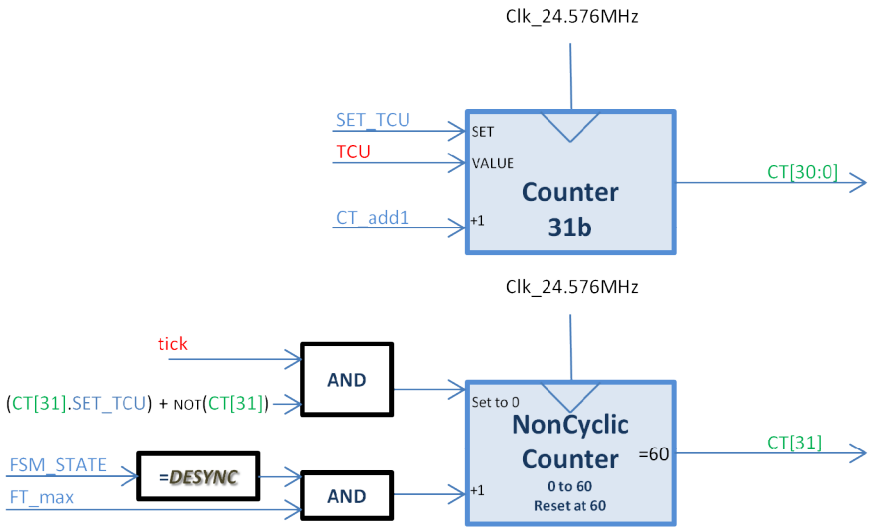


Figure 10: COARSE TIME COUNTER

2.3.3.1.2 Software Interface

Address	Name	Description
0x00	CONTROL	Permit to generate a tick by software. Bit: Description 31..3: Not Used 2: LFR SubSystem soft Reset 1: Time Management Software Reset 0: Time Management Software tick.
0x04	COARSE_TIME_new	If modified, the next value of the coarse time. Bit: Description 31: Unused 30..0: Coarse Time Next
0x08	COARSE_TIME	Coarse Time. Bit: Description 31: UnSychronized 30..0: Coarse Time
0x0C	FINE_TIME	Coarse Time. Bit: Description 31..16: Not used 15..0: Fine Time

2.3.3.1.3 Operations

The nominal behaviour is the synchronized mode (see Figure 11). In this case the time management receives a tick each second. Each time that a new tick is received, the coarse time is updated (previous+1 or TCU) and the fine time is reset to 0.

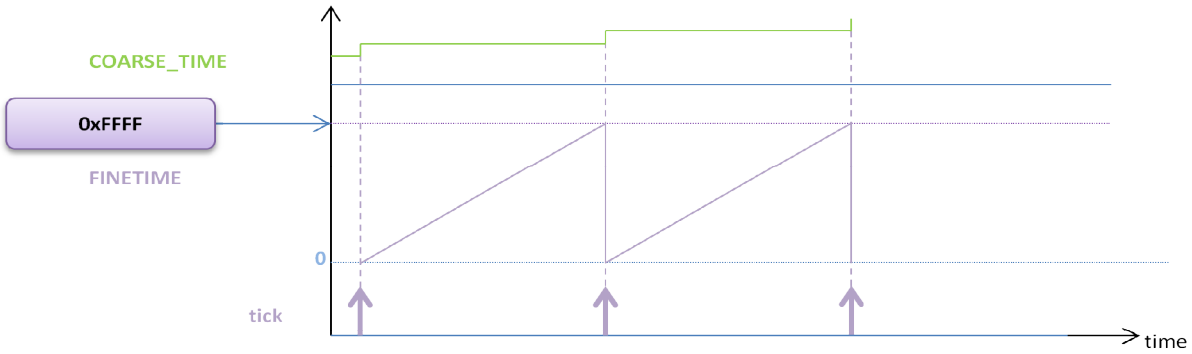


Figure 11: Synchronized operations

If the tick signal is not completely synchronous (see Figure 12), there is a little positive or negative variations of the seconds. In the first case, the fine time counter arrives to the max value before the reception the tick. The FINE_TIME and COARSE_TIME value are not changed while the new tick is not received. In the second case, the tick is received before the FINE_TIME reaches the max value, the coarse time is updated and the fine time is reset.

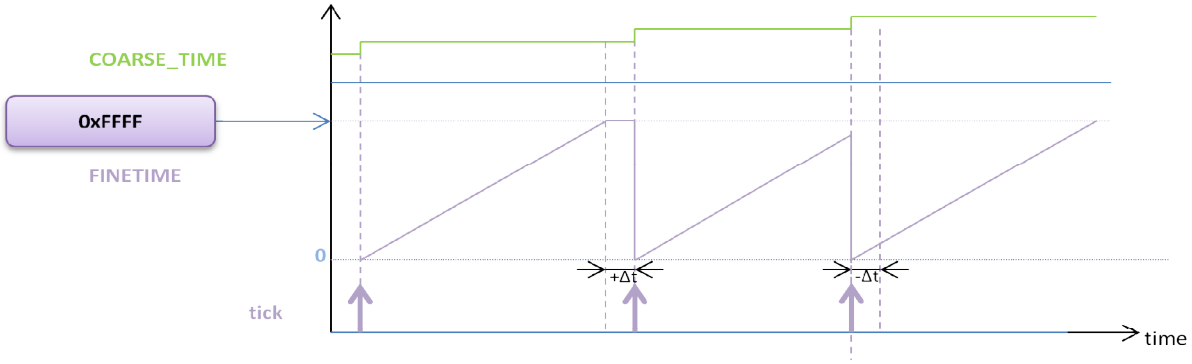


Figure 12: Synchronized operations with a tick almost synchronous

If the delta delay is greater than a limit fixed to 1ms, the module is considering to be in DESYNC mode (see Figure 13). In this case, FINE_TIME and COARSE_TIME are not updated while the 1 ms delay is not elapsed. After that, the COARSE_TIME is increased by one and FINE_TIME is set to 1ms.

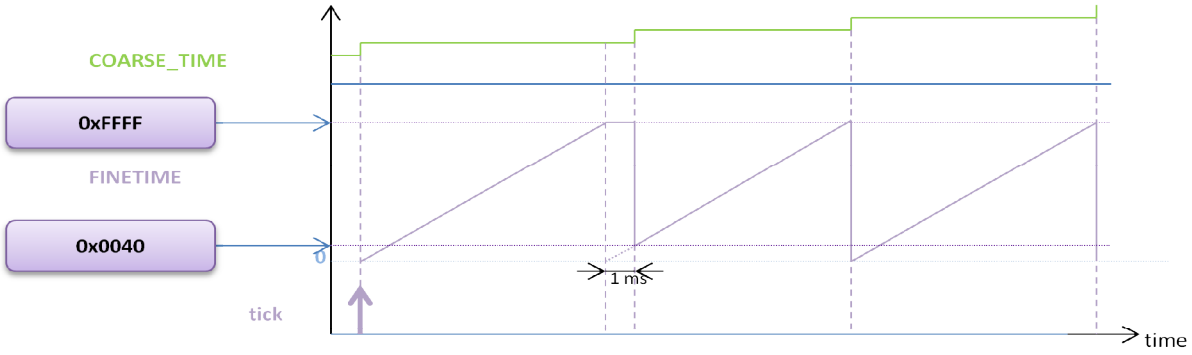


Figure 13: Synchronized to Desynchronized operations

Once the system is in the desynchronized state, the COARSE_TIME is increased by one and FINE_TIME is reset to 0, each time that FINE_TIME reaches the max value.

If a tick occurs when the system is in the desynchronized state, the fine time is reset to 0 and the coarse time is updated (See Figure 14). There are 3 options to update it:

- There is a new TCU, the next coarse time is set to this value,
- The tick occurs when the FINE_TIME value is under half of the max value, then the COARSE_TIME is not changed,
- The tick occurs when the FINE_TIME value is greater than half of the max value, then the COARSE_TIME value is increased by one.

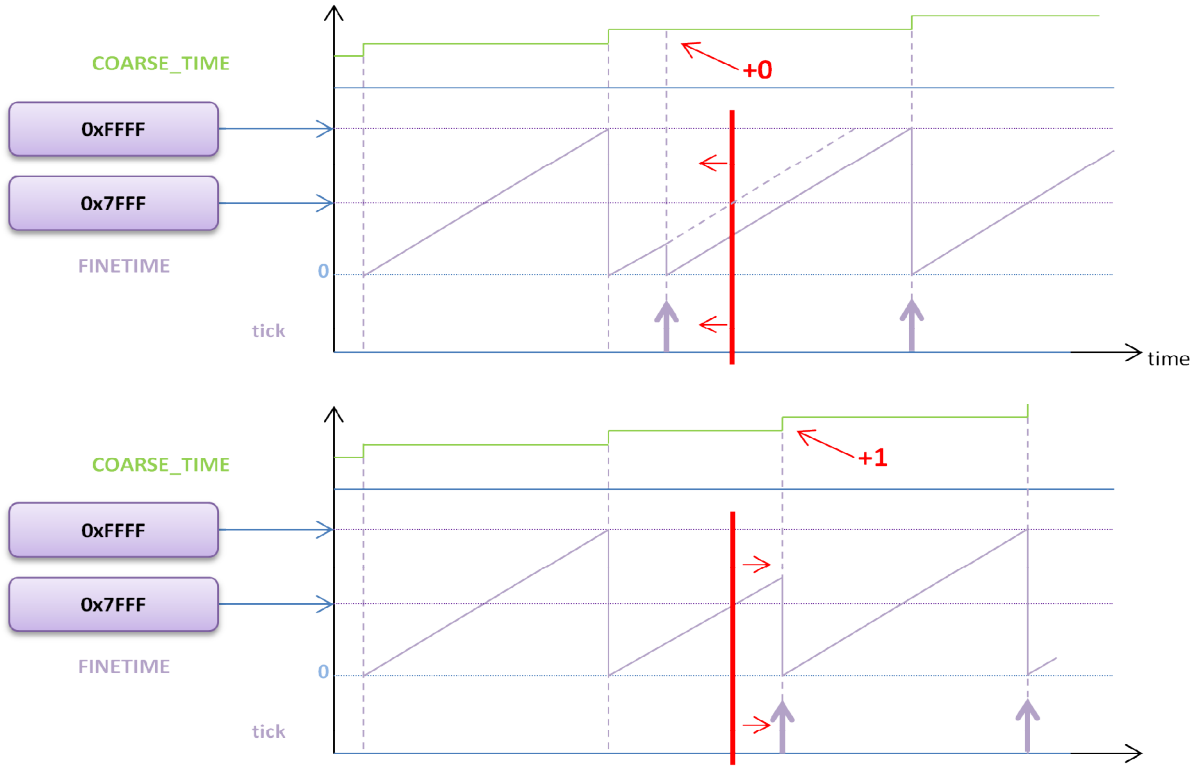
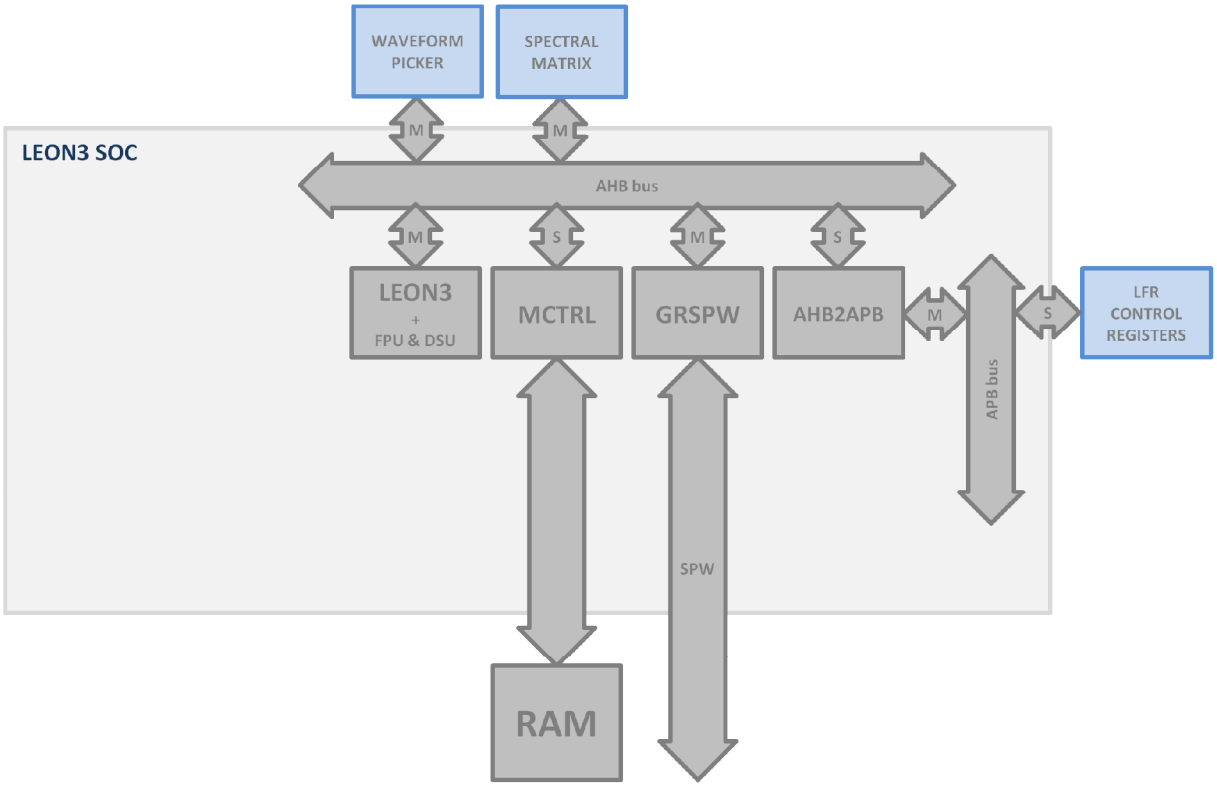


Figure 14: Desynchronized to synchronized operations

It is important no notice that during the transition from a desynchronized state to a synchronized state and in the case where there is no new TCU and the FINE_TIME is below half the max value, the time **decreases**.

2.3.4 Leon3 SoC

2.3.4.1 Block View



2.3.4.2 AHB/APB Mapping

Table 1 : AHB MASTER LIST

AHB Index	Name	Descritption
0	Leon3	Leon3 processor
1	GRSPW	Gaisler Space Wire IP.
2	WaveForm Picker	LPP WaveForm Picker IP.
3	Matrix Spectral	LPP Matrix Spectral IP.
4	UART	Gaisler AHB UART.

Warning: Gaislerôs AHB UART must be the last Master.

Table 2 : AHB/APB Slave List and interruption

Name	BASE ADDRESS	LENGTH	Slave Type	Index	IRQ	Descritption
DSU3	0x90000000	0x0FFFFFFF	AHB	2		Gaisler Debug System Unit
APBCTRL	0x80000000	0x0FFFFFFF	AHB	1		APB Controller
MCTRL	0x00000000	0x0FFFFFFF	AHB	0		Memory Controller
	0x80000000	0x000000FF	APB	0		
IRQMP	0x80000200	0x000000FF	APB	2		Interrupt controller
APB_LFR_Time_Managment	0x80000600	0x000000FF	APB	6	12	Time management module developed by LPP
LPP_LFR	0x80000F00	0x000000FF	APB	15	6,14	LPP LFR registersĉ for controlled

						and observed Waveform Picker and Spectral Matrix Subsystems
Apb_Uart	0x80000100	0x000000FF	APB	1	2	Uart connected on APB bus
GPTimer	0x80000300	0x000000FF	APB	3	8	Timer module

2.3.4.3 LFR Controlôs Registers

Tableau 3 : LPP_LFR Registers List

Address	Name	Description
0x80000F00	SPECTRAL_MATRIX_CONFIG	Permit to activate the interruption from LFR_SM module and run the Spectral Matrix. Bit: Description 31..3: Not Used 2: Run the Spectral matrix 1: Active the interruption when a new Matrix is completely sent in Memory. 0: Active the interruption when a error occurs.
0x80000F04	SPECTRAL_MATRIX_STATUS	Indicates the status of LFR_SM module. A write must be done after a read to ñeraseò the status. Bit: Description 31..11: Not Used 10: Set when an error ñwrite into Full FIFO_2ò occurs 9: Set when an error ñwrite into Full FIFO_1ò occurs 8: Set when an error ñwrite into Full FIFO_0ò occurs 7: Set when an error ñBuffer Fullò occurs 6: Not Used 5: Set when a matrix f2 is ready into buffer 1 4: Set when a matrix f2 is ready into buffer 0 3: Set when a matrix f1 is ready into buffer 1 2: Set when a matrix f1 is ready into buffer 0 1: Set when a matrix f0 is ready into buffer 1 0: Set when a matrix f0 is ready into buffer 0
0x80000F08	SPECTRAL_MATRIX_ADDRESS_F0_0	Base address of the first buffer where the Spectral Matrix must write the next Spectral Matrix F0
0x80000F0C	SPECTRAL_MATRIX_ADDRESS_F0_1	Base address of the second buffer where the Spectral Matrix must write the next Spectral Matrix F0
0x80000F10	SPECTRAL_MATRIX_ADDRESS_F1_0	Base address of the first buffer where the Spectral Matrix must write the next Spectral Matrix F1
0x80000F14	SPECTRAL_MATRIX_ADDRESS_F1_1	Base address of the second buffer where the Spectral Matrix must write the next Spectral Matrix F1
0x80000F18	SPECTRAL_MATRIX_ADDRESS_F2_0	Base address of the first buffer where the Spectral Matrix must write the next Spectral Matrix F2
0x80000F1C	SPECTRAL_MATRIX_ADDRESS_F2_1	Base address of the second buffer where the Spectral Matrix must write the next Spectral Matrix F2
0x80000F20	SPECTRAL_MATRIX_COARSETIME_F0_0	CoarseTime of the first data of the first buffer Spectral Matrix F0
0x80000F24	SPECTRAL_MATRIX_FINETIME_F0_0	FineTime of the first data of the first buffer Spectral Matrix F0
0x80000F28	SPECTRAL_MATRIX_COARSETIME_F0_1	CoarseTime of the first data of the 2nd buffer Spectral Matrix F0
0x80000F2C	SPECTRAL_MATRIX_FINETIME_F0_1	FineTime of the first data of the 2nd buffer Spectral Matrix F0
0x80000F30	SPECTRAL_MATRIX_COARSETIME_F1_0	CoarseTime of the first data of the first buffer Spectral Matrix F1

0x80000F34	SPECTRAL_MATRIX_FINETIME_F1_0	FineTime of the first data of the first buffer Spectral Matrix F1
0x80000F38	SPECTRAL_MATRIX_COARSETIME_F1_1	CoarseTime of the first data of the 2nd buffer Spectral Matrix F1
0x80000F3C	SPECTRAL_MATRIX_FINETIME_F1_1	FineTime of the first data of the 2nd buffer Spectral Matrix F1
0x80000F40	SPECTRAL_MATRIX_COARSETIME_F2_0	CoarseTime of the first data of the first buffer Spectral Matrix F2
0x80000F44	SPECTRAL_MATRIX_FINETIME_F2_0	FineTime of the first data of the first buffer Spectral Matrix F2
0x80000F48	SPECTRAL_MATRIX_COARSETIME_F2_1	CoarseTime of the first data of the 2nd buffer Spectral Matrix F2
0x80000F4C	SPECTRAL_MATRIX_FINETIME_F2_1	FineTime of the first data of the 2nd buffer Spectral Matrix F2
0x80000F50	SPECTRAL_MATRIX_LENGTH	Length of each spectral matrix buffer
0x80000F54	WAVEFORM_PICKER_DATASHAPING	Permit to set the data shaping parameter. Bit / Description 31..6 / Not Used 5 / Set the data shaping Parameter R2. 4 / Set the data shaping Parameter R1. 3 / Set the data shaping Parameter R0. 2 / Set the data shaping Parameter SP1. 1 / Set the data shaping Parameter SP0. 0 / Set the data shaping Parameter BW.
0x80000F58	WAVEFORM_PICKER_CONTROL	Permit to control activation of each channel and to configure the burst mode for the concerning channel. Bit / Description 31..7 / Not Used 6 / Enable Burst mode for Data f2 5 / Enable Burst mode for Data f2 4 / Enable Burst mode for Data f2 3 / Enable acquisition of Data f3 2 / Enable acquisition of Data f2 1 / Enable acquisition of Data f1 0 / Enable acquisition of Data f0
0x80000F5C	WAVEFORM_PICKER_ADDRESS_F0_0	Base address of the 1rst buffer where the Waveform Picker must write the next Data f0
0x80000F60	WAVEFORM_PICKER_ADDRESS_F0_1	Base address of the 2nd buffer where the Waveform Picker must write the next Data f0
0x80000F64	WAVEFORM_PICKER_ADDRESS_F1_0	Base address of the 1rst buffer where the Waveform Picker must write the next Data f1
0x80000F68	WAVEFORM_PICKER_ADDRESS_F1_1	Base address of the 2nd buffer where the Waveform Picker must write the next Data f1
0x80000F6C	WAVEFORM_PICKER_ADDRESS_F2_0	Base address of the 1rst buffer where the Waveform Picker must write the next Data f2
0x80000F70	WAVEFORM_PICKER_ADDRESS_F2_1	Base address of the 2nd buffer where the Waveform Picker must write the next Data f2
0x80000F74	WAVEFORM_PICKER_ADDRESS_F3_0	Base address of the 1rst buffer where the Waveform Picker must write the next Data f3
0x80000F78	WAVEFORM_PICKER_ADDRESS_F3_1	Base address of the 2nd buffer where the Waveform Picker must write the next Data f3
0x80000F7C	WAVEFORM_PICKER_STATUS	Indicates the status of LFR_Waveform module Bit / Description 31..16 / Not Used

		15..12	Vector of New Error bits for Data f0 to f3. The bit 0 is for new error of data f0, etc. Set when internal buffer (FIFO) is full and a new data should be writing. Those bits are automatically reset after read.
		11..8	Vector of Full Error bits for Data buffer f0_0 to f3_0. The bit 0 is for full error of data f0_0, etc. Set when Waveform Buffer of Data is full and that a new data should be writing. Those bits are automatically reset after read.
		7..0	Vector of Full Status bits for each data buffer f0_0 to f3_1. Set when Waveform Buffer of Data is full. Each bit is automatically reset after read.
0x80000F80	WAVEFORM_PICKER_DELTASNAPSHOT	DeltaSnapshot parameter.	
0x80000F84	WAVEFORM_PICKER_DELTA_f0	Delta_f0 parameter.	
0x80000F88	WAVEFORM_PICKER_DELTA_f0_2	Delta_f0_2 parameter.	
0x80000F8C	WAVEFORM_PICKER_DELTA_f2	Delta_f2 parameter.	
0x80000F90	WAVEFORM_PICKER_DELTA_f1	Delta_f1 parameter.	
0x80000F94	WAVEFORM_PICKER_NBDATABYBUFFER	Number of data by buffer (one data is 6*2B)	
0x80000F98	WAVEFORM_PICKER_NBSNAPSHOT	Nb_snapshot_param parameter.	
0x80000F9C	WAVEFORM_PICKER_START_DATE	The start date. When this date is equal or lesser than the time management date, the waveform starts.	
0x80000FA0	WAVEFORM_PICKER_COARSETIME_F0_0	CoarseTime of the first data of the first WaveFormPicker buffer F0	
0x80000FA4	WAVEFORM_PICKER_FINETIME_F0_0	FineTime of the first data of the first WaveFormPicker buffer F0	
0x80000FA8	WAVEFORM_PICKER_COARSETIME_F0_1	CoarseTime of the first data of the 2nd WaveFormPicker buffer F0	
0x80000FAC	WAVEFORM_PICKER_FINETIME_F0_1	FineTime of the first data of the 2nd WaveFormPicker buffer F0	
0x80000FB0	WAVEFORM_PICKER_COARSETIME_F0_0	CoarseTime of the first data of the first WaveFormPicker buffer F0	
0x80000FB4	WAVEFORM_PICKER _FINETIME_F0_0	FineTime of the first data of the first WaveFormPicker buffer F0	
0x80000FB8	WAVEFORM_PICKER_COARSETIME_F0_1	CoarseTime of the first data of the 2nd WaveFormPicker buffer F0	
0x80000FBC	WAVEFORM_PICKER _FINETIME_F0_1	FineTime of the first data of the 2nd WaveFormPicker buffer F0	
0x80000FC0	WAVEFORM_PICKER_COARSETIME_F0_0	CoarseTime of the first data of the first WaveFormPicker buffer F0	
0x80000FC4	WAVEFORM_PICKER _FINETIME_F0_0	FineTime of the first data of the first WaveFormPicker buffer F0	
0x80000FC8	WAVEFORM_PICKER_COARSETIME_F0_1	CoarseTime of the first data of the 2nd WaveFormPicker buffer F0	
0x80000FCC	WAVEFORM_PICKER _FINETIME_F0_1	FineTime of the first data of the 2nd WaveFormPicker buffer F0	
0x80000FD0	WAVEFORM_PICKER_COARSETIME_F0_0	CoarseTime of the first data of the first WaveFormPicker buffer F0	
0x80000FD4	WAVEFORM_PICKER _FINETIME_F0_0	FineTime of the first data of the first WaveFormPicker buffer F0	

<table>
<tr><td>0x80000FD8</td><td>WAVEFORM_PICKER_COARSETIME_F0_1</td><td>CoarseTime of the first data of the 2nd WaveFormPicker buffer F0</td></tr>
<tr><td>0x80000FDC</td><td>WAVEFORM_PICKER _FINETIME_F0_1</td><td>FineTime of the first data of the 2nd WaveFormPicker buffer F0</td></tr>
<tr><td>0x80000FE0</td><td>WAVEFORM_PICKER_ BUFFER_LENGTH</td><td>Nb_word_by_buffer parameter. Indicates the bufferôs size in words (4B)</td></tr>
<tr><td>0x80000FF0</td><td>LFR_RTL_VERSION</td><td>This is the version of RTL.
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>31..24</td><td>Not Used</td></tr>
<tr><td>23..16</td><td>Current board
0 => mini-LFR
1 => LFR-em</td></tr>
<tr><td>15..8</td><td>Current major version</td></tr>
<tr><td>7..0</td><td>Current minor version</td></tr>
</table></td></tr>
</table>

2.3.5 LPP_LFR_FILTER

2.3.5.1 Block View

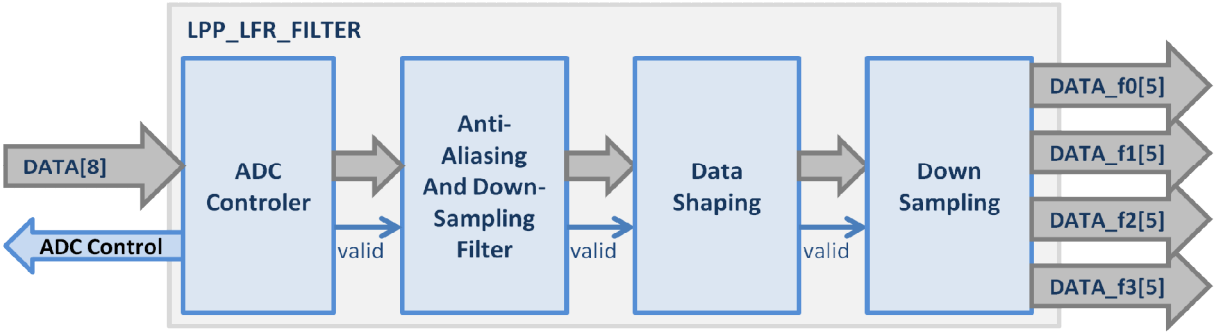


Figure 15 LPP_LFR_FILTERôs block view

2.3.5.2 ADC Controller

The ADC controller is interface to the 8 ADC RHF1401 as shown in Figure 16. Acquisition signal (CNV) and clock are the same for all ADC. The DATA from ADC are demuxed in function of the Ouput Enable signal.

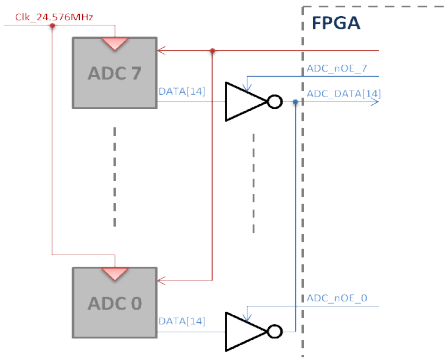


Figure 16 ADC interface

Acquisition signal (CNV) is generated into the FPGA by CNV_GEN block. This block is a simple counter on CLK_24576MHz. The CNV is a cyclic signal with a frequency of 24.576MHz/25 = 983,04kHz and a duty cycle of 13/25.

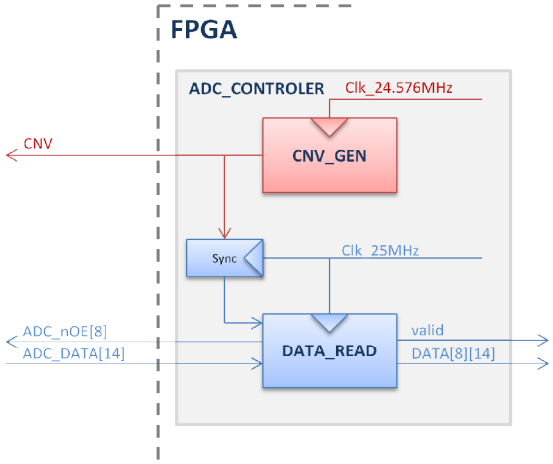


Figure 17 ADC interface

Once the acquisition is done (CNV is reset), the data are ready into the ADC. The DATA_READ block read one by one the data. For each channel, a simple moving average is compute with the

previous data. Each ten data load, the 8 average data are sent in parallel to the Anti-Aliasing and Downsampling Filter. The data are output at the frequency 983.04kHz/10 = 98.304 kHz.

2.3.5.3 Anti-aliasing and Downsampling Filter

The anti-aliasing and downsampling filter component is based on the VHDL component IIR_CEL_CTRLR_v2 developed at LPP and documented in [RD2].

The filter receives 8 channels of samples at 98 304 Hz, and produces 8 channels of samples filtered at f0 = 24 576 Hz.

For each channel, the filter is a 5 stages IIR filter. Each stage is a second order section which computes the following formula:

$$y[n] = a1 . y[n-1] + a2 . y[n-2] + b0 . x[n] + b1 . x[n-1] + b2 . x[n-2],$$

y[n] being the output of the stage and x[n] being the input of the cell.

All coefficients (a1, a2, b0, b1 and b2) are stored in the FPGA and cannot be modified after the instantiation of the component. One set of coefficients is used per stage.

The Matlab tool ñFilter Design and Analysisò has been used to compute the coefficients of the filter. After the computation, the coefficients are multiplied by 128 and resized as 9 bits signed integers before being written in a dedicated VHDL file used by the VHDL design tool at the instantiation of the VHDL component. The coefficients used for the LFR input down sampling filter are the following:

	b0	b1	b2	a1	a2
Stage 1	58	-66	58	189	-111
Stage 2	58	-57	58	162	-81
Stage 3	29	-17	29	136	-55
Stage 4	15	4	15	114	-33
Stage 5	15	24	15	100	-20

Before being fed into the IIR_CEL_CTRLR_v2 component, the samples are resized to 18 bits signed integers by adding two extra bits to the initial 16 bits and by propagating the sign bit leftward. The position of the radix point is 7 thus the coefficients initially computed by Matlab, which values are all below 1, have been multiplied by 2^7 = 128. Inside the filter, each multiplication performed by the Arithmetical and Logical Unit between a coefficient (9 bits) and a sample (18 bits) results in a data coded on 27 bits. Only the 18 most significant bits are kept for the next calculation. The filter has a slight attenuation that has been added after having preformed several tests on the VHDL component using ModelSim. The objective being to avoid integer overflows.

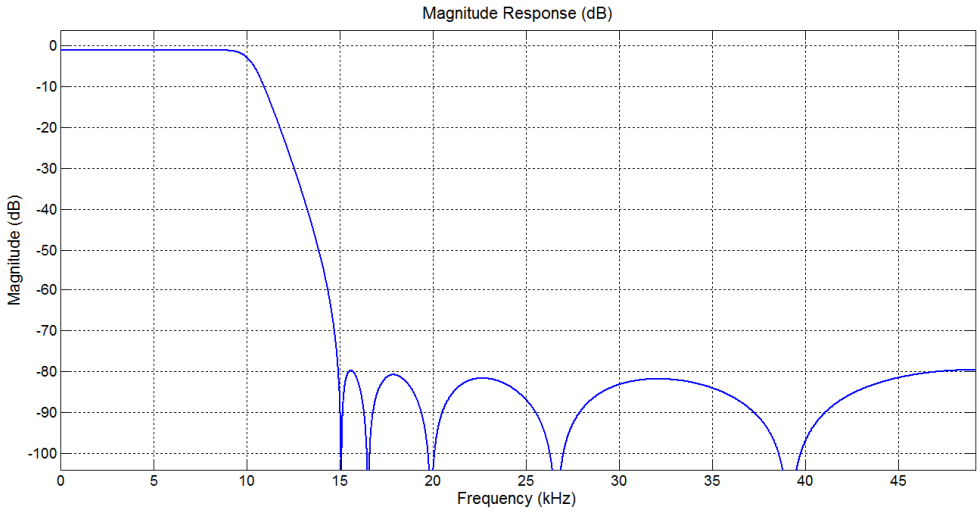


Figure 18 Transfer function of the LFR input anti-aliasing filter (98 304 Hz => 24 576 Hz)

2.3.5.4 Data Shaping

The data shaping is a simple manipulation of the data between the Anti-Aliasing and downsampling filter and the DownSampling Block. Either the channel 1(respectively 2) is not modify or it is replaced by the difference beteween channel 1 and 0 (respectively 2 and 1). This choice is make by configuring the register SP0 (respectively SP1).

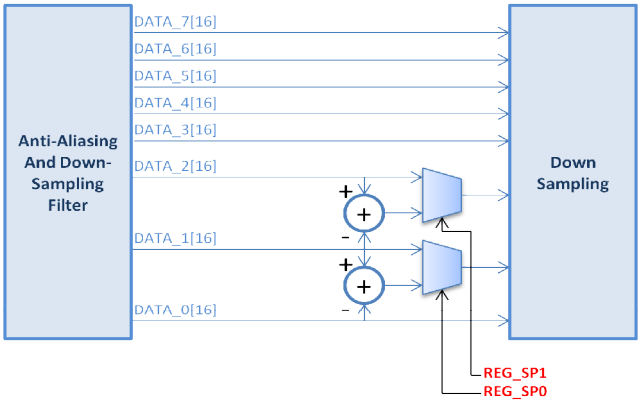


Figure 19 Data Shaping

2.3.5.5 Downsampling Filter

The downsampling filter component is based on the VHDL component IIR_CEL_CTRLR_v2 and LFR_CIC developed at LPP and documented in [RD2].

The block receives 8 channels of samples at 24 576 Hz, and produces 4 lane of 6 channels of samples filtered at f0 = 24 576 Hz, f1 = 4 096Hz, f2 = 256 Hz and f3 = 16Hz.

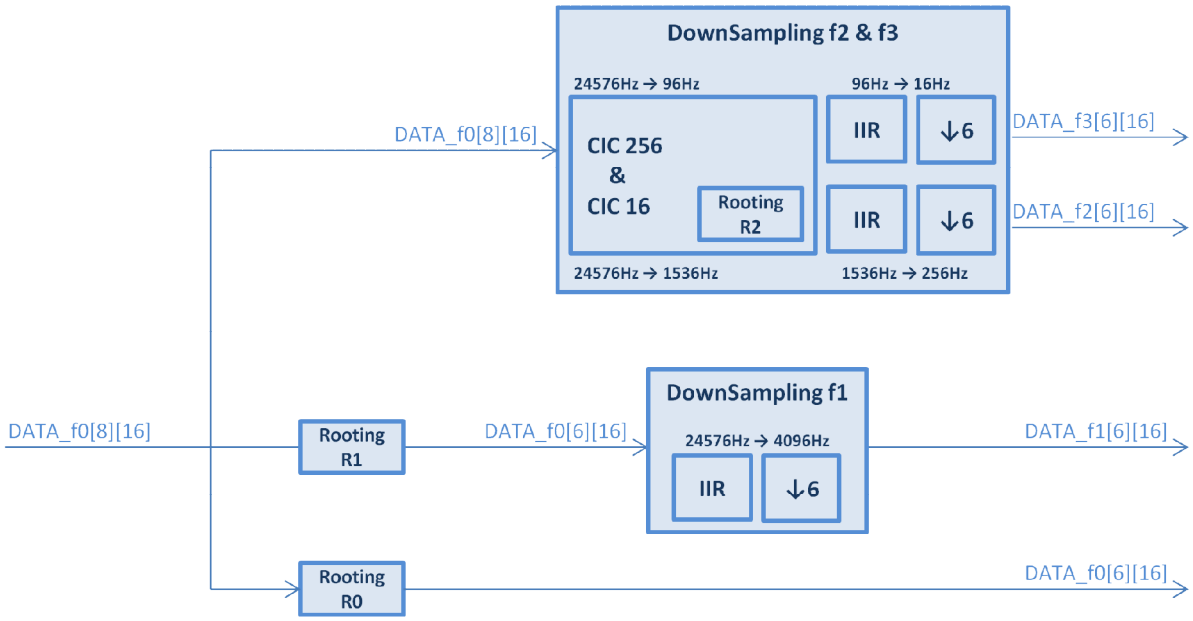


Figure 20 DownSampling Filter

Each routing module is a simple selection of the 8 input channels in function of the register parameter Ri (i Ɑ{0; 1; 2}).

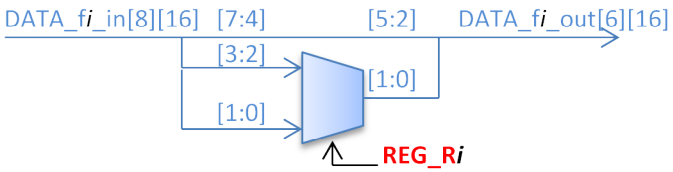


Figure 21 Routing module

The lane f0 is just the input lane passed through the Rooting module R0.

For the generation of the data lane f1, the input lane passes through the routing module R1. The 6 channels of samples outgoing are filtering with an IIR_CEL_CTRLR_v2 module. This module is configured to be a Low Pass Band with a cut frequency of 4 kHz. The result is down sampled by 6 to obtain a data frequency of 4096Hz.

For the generation of the data lane f2 and f3, the 8channels input passes are filtering with a LFR_CIC module. The LFR_CIC generate 2 lane. The first lane is equivalent to the output of CIC with a down sampling factor 16. And the second lane is equivalent to the output of a CIC with a downsampling factor 256. Each output lane of the LFR_CIC is followed by an IIR_CEL_CTRLR_v2 module. Each module is configured to be a Low Pass Band with a cut frequency equal to the input frequency divided by 6. The result is down sampled by 6 to obtain a data frequency of 256Hz and 16Hz.

2.3.6 LPP_LFR_WAVEFORM

2.3.6.1 Block View

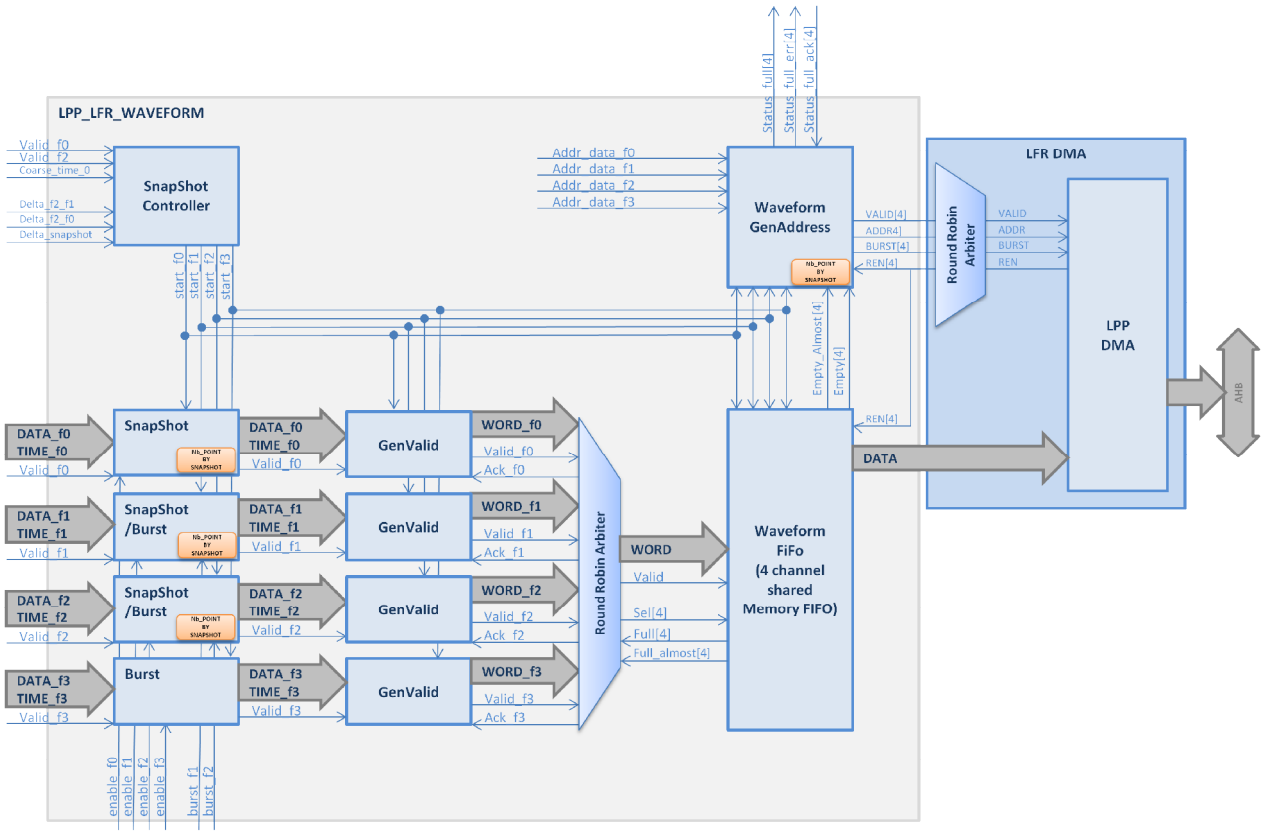


Figure 22 : Waveform Picker's Block diagram view

As shown in Figure 22, the waveform picker is a block which is composed of 4 ñstagesò: SnapShot, GenValid, Waveform FIFO and Waveform Gen Address (interfaced with the LFR DMA).

The snapshot stage receives 4 data channels. It is controlled by the snapshot controller. For each data being received, there is a valid bit which indicates if the current data is a new valid data and a time value which indicates the date of this data (the date when the data was received into the FPGA). The snapshot controller according to an internal counter decides whether the current data should be saved (in snapshot mode) or not. The Snapshot block, depending on the configuration (enable, enable burst and start from controller), sends the couple (data, time) and valid bit to the second stage GenValid. When a channel is in Snapshot mode, a counter count the valid data received since the last ñstartò. If the counter value is not equal to the ñNB_POINT_BY_SNAPSHOTò when the next start occurred, an errors signal is raised (STATUS_SNAPSHOT_ERR).

The GenValid stage registers the couple (data, time). This couple is transmitted word by word into the FIFO. Just the time of the first couple after the reception of the start is written into the FIFO. When all the data (3 WORD) are written into the FIFO, a new couple (data, time) can be received. If a new data is received before, an error occurs (STATUS_GEN_VALID_ERR).

A round robin arbiter selects one of the channels between the GenValid stage and the WaveForm FIFO. A channel can be selected only if there is a valid couple (data, time) and if there is at least 5 WORDs free into the corresponding FIFO (i.e. the almost full signal is not asserted).

The third block is Waveform FIFO. This FIFO is a 4 channel shared Memory FIFO. There is one channel for each frequency. The data received by LPP_LFR_WAVEFORM are composed of the TIME (32b+16b) and sample DATA (6*16b). The WaveForm FIFO is read by the LFR_DMA. There is a debug register ñSTATUS_WFP_FIFO_MAX_OCCUPANCYò which indicates the Maximum Number of Data in the FIFO since the start of the FPGA. This value should not be higher than 18 (15+3).

The GEN_ADDRESS is the last Waveform Picker block. This block indicates for each channel, the current address pointer to the DMA and if the DMA must write data using a Single or a Burst access. The GEN_ADDRESS counts the number of data written by LFR_DMA. When the number of data is equal to 2+3* NB_POINT_BY_SNAPSHOT, a status full is sent into the LFR_DMA. The software (executed on LEON3) must then send a new START_ADDRESS and reset the status full register. If a start is sent by Snapshot controller before, an error occurs (STATUS_GEN_ADDRESS_ERR).

2.3.6.2 Generics Parameters

Name	Description	Type	Default
Hindex	AHB index on the Leon3 SoC AHB Bus	Integer	2
Tech	Type of FPGA technology (See Leon3 doc for more details)	Integer	0 (inferred)
data_size	Number of data bits	Integer	160
NB_POINT_BY SNAPSHOT_SIZE	Bitôs number of NB_POINT_BY_SNAPSHOT	Integer	11
Delta_snapshot_size	Bitôs number of Delta_snapshot	Integer	11
Delta_vector_size	Bitôs number of Delta parameter	Integer	20
Delta_vector_size_f0_2	Bitôs number of Delta_f0_2	Integer	7

2.3.6.3 Interface

Name	Description	Active	Direction	Size
Clk	Clock	N/A	Input	1
Rstn	Reset	Low	Input	1
AHB_master_in	AHB master input interface (see AHB_Mst_In_Type into Leon3 documentation)	N/A	Input	N/A
AHB_master_out	AHB master output interface (see AHB_Mst_Out_Type into Leon3 documentation)	N/A	Output	N/A
Coarse_time_0	Lower bit of the Coarse Time Signal. The Coarse time signal indicates the ñtimeò in second and its lower bits changes each second.	N/A	Input	1
Delta_snapshot	Number of seconds (Coarse_Time_0 transition) to wait between 2 snapshot	N/A	Input	Delta_snapshot_size
Delta_f2_f1	Number of T0=1/f0 period to wait between snapshotôs start at f2 and snapshotôs start of the data at ñfrequency f1ò	N/A	Input	Delta_f2_f1_size
Delta_f2_f0	Number of T0=1/f0 period to wait between snapshotôs start f2 and snapshotôs start of the data at ñfrequency f0ò	N/A	Input	Delta_f2_f0_size

Enable_f0	Enable snapshot of data at ñfrequency f0ò	High	Input	1
Enable_f1	Enable snapshot of data at ñfrequency f1ò	High	Input	1
Enable_f2	Enable snapshot of data at ñfrequency f2ò	High	Input	1
Enable_f3	Enable snapshot of data at ñfrequency f3ò	High	Input	1
Burst_f0	Enable burst mode of snapshot for the data at ñfrequency f0ò	High	Input	1
Burst_f1	Enable burst mode of snapshot for the data at ñfrequency f1ò	High	Input	1
Burst_f2	Enable burst mode of snapshot for the data at ñfrequency f2ò	High	Input	1
Nb_burst_available	Number of burst ñ16Bò which can be writing into a waveform buffer. Typically: WavefromBufferSize (in B) /16B.	N/A	Input	Nb_burst_available_size
Nb_snapshot_param	Number of points saved during a snapshot.	N/A	Input	Nb_snapshot_param_size
Status_full	Ith bit is set when waveform picker has handled a complete series of data at the ñfrequency fiò, this bit is unset when ith bit of status_full_ack is set.	High	Output	4
Status_full_ack	When the Ith bit is set, waveform picker can continue to handle the data at ñfrequency fiò and the ith bit of status_full can be unset.	High	Input	4
Status_full_err	Ith bit is set when waveform picker receives a new data at ñfrequency fiò and ith bit of status_full is set.	High	Input	4
Addr_data_f0	Waveformôs subsystem writes into external memory (through the AHB bus) the data at frequency f0. Addr_data_f0 is the start address of memory space reserved for.	N/A	Input	32
Addr_data_f1	Idem than addr_data_f0 but for data at frequency f1.	N/A	Input	32
Addr_data_f2	Idem than addr_data_f0 but for data at frequency f2.	N/A	Input	32
Addr_data_f3	Idem than addr_data_f0 but for data at frequency f3.	N/A	Input	32
Data_f0_in	Sample Data at frequency f0	N/A	Input	data_size
Data_f1_in	Sample Data at frequency f1	N/A	Input	data_size
Data_f2_in	Sample Data at frequency f2	N/A	Input	data_size
Data_f3_in	Sample Data at frequency f3	N/A	Input	data_size
Data_f0_in_valid	Valid bit for data at frequency f0.	High	Input	data_size

	When this bit is set, the Sample Data is a new valid data and can be used.			
Data_f1_in_valid	Idem at f1	High	Input	data_size
Data_f2_in_valid	Idem at f2	High	Input	data_size
Data_f3_in_valid	Idem at f3	High	Input	data_size

2.3.6.4 Operations

2.3.6.4.1 Software Interface

The table below lists the APB registers in APB LPP_LFR Registers used to control and observe the WaveForm Picker subsystem.

Tableau 4 : WaveForm Picker Registers List

Address	Name	Description
0x80000F50	WAVEFORM_PICKER_DATASHAPING	Permit to set the data shaping parameter. Bit: Description 31..6: Not Used 5: Set the data shaping Parameter R2. 4: Set the data shaping Parameter R1. 3: Set the data shaping Parameter R0. 2: Set the data shaping Parameter SP1. 1: Set the data shaping Parameter SP0. 0: Set the data shaping Parameter BW.
0x80000F54	WAVEFORM_PICKER_CONTROL	Permit to control activation of each channel and to configure the burst mode for the concerning channel. Bit: Description 31..7: Not Used 6: Enable Burst mode for Data f2 5: Enable Burst mode for Data f2 4: Enable Burst mode for Data f2 3: Enable acquisition of Data f3 2: Enable acquisition of Data f2 1: Enable acquisition of Data f1 0: Enable acquisition of Data f0
0x80000F58	WAVEFORM_PICKER_ADDRESS_F0	Base address of the buffer where the Waveform Picker must write the next Data f0
0x80000F5C	WAVEFORM_PICKER_ADDRESS_F1	Base address of the buffer where the Waveform Picker must write the next Data f1
0x80000F60	WAVEFORM_PICKER_ADDRESS_F2	Base address of the buffer where the Waveform Picker must write the next Data f2
0x80000F64	WAVEFORM_PICKER_ADDRESS_F3	Base address of the buffer where the Waveform Picker must write the next Data f3
0x80000F68	WAVEFORM_PICKER_STATUS	Indicates the status of LFR_Waveform module Bit: Description 31..12: Not Used 11..8: Vector of New Error bits for Data f0 to f3. The bit 0 is for new error of data f0, etc. Set when internal buffer (FIFO) is full and a new data should be writing. Those bits are automatically reset after read. 7..4: Vector of Full Error bits for Data f0 to f3. The bit 0 is

		for full error of data f0, etc. Set when Waveform Buffer of Data is full and that a new data should be writing. Those bit are automatically reset after read. 3..0 Vector of Full Status bits for Data f0 to f3. The bit 0 is for full buffer of data f0, etc. Set when Waveform Buffer of Data is full. Those bit are automatically reset after read.
0x80000F6C	WAVEFORM_PICKER_DELTASNAPSHOT	DeltaSnapshot parameter.
0x80000F70	WAVEFORM_PICKER_DELTA_f0	Delta_f0 parameter.
0x80000F74	WAVEFORM_PICKER_DELTA_f0_2	Delta_f0_2 parameter.
0x80000F78	WAVEFORM_PICKER_DELTA_f1	Delta_f1 parameter.
0x80000F7C	WAVEFORM_PICKER_DELTA_f2	Delta_f2 parameter.
0x80000F80	WAVEFORM_PICKER_NBDATABYBUFFER	Number of data by buffer (one data is 6*2B)
0x80000F84	WAVEFORM_PICKER_NBSNAPSHOT	Nb_snapshot_param parameter.
0x80000F88	WAVEFORM_PICKER_START_DATE	The start date. When this date is equal or lesser than the time management date, the waveform starts.
0x80000F8C	WAVEFORM_PICKER_NBWORDBY BUFFER	Nb_word_by_buffer parameter. Indicates the bufferôs size in words (4B)

2.3.6.4.1.1 Snapshot configuration

The time intervals between the snapshots (T_{WF} into the RPW-MEB-LFR-00003 documentation) can be configured by writing the value in the WAVEFORM_PICKER_DELTASNAPSHOT register (in seconds).

As shown in Figure 23, a decounter is initialized with the delta_snapshot value. Each second (coarse_time_0 front from the Time Management IP), the decounter counts down. When the decounter is null, a start_snapshot_f2 is launched and the decounter is reinitialized with the delta_snapshot value.

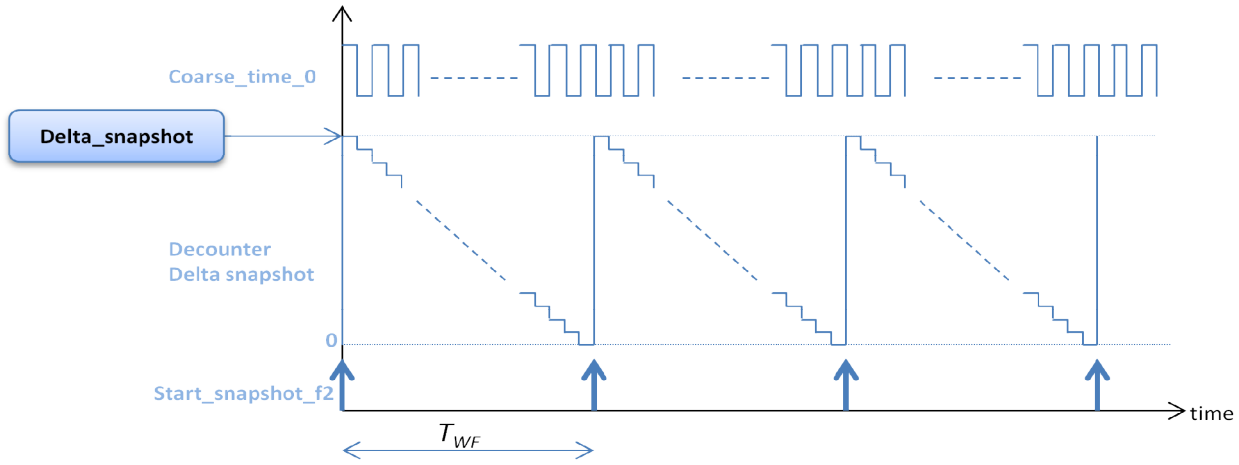


Figure 23 : Start Snapshot Generation for the Data at the frequency f2

As shown in Figure 24, a decounter is initialized at each start_snapshot_f2 event with the delta_f2_f0 value. This decounter counts down until 0 on each valid_data_f0. When its value is equal to delta_f2_f1, a start_snapshot_f1 event is generated, and when it is equal to 0 a start_snapshot_f0 event is generated.

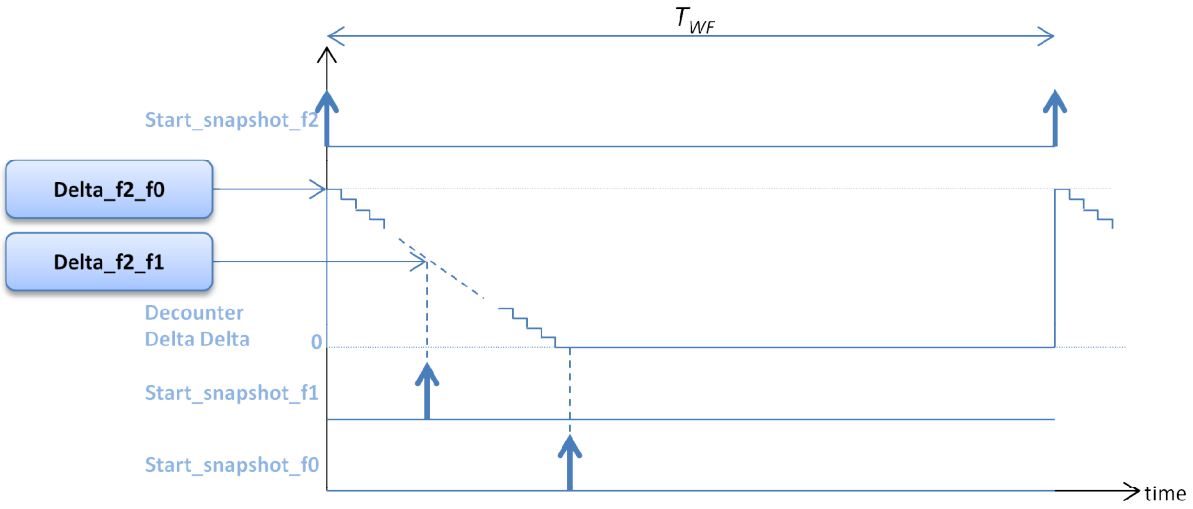


Figure 24 : Start Snapshot Generation for the Data at the frequency f1 and f0

As shown in Figure 25, for each channel, when start_snapshot is asserted a snapshot starts. The number of points by snapshot is configurable with the parameter Nb_snapshot_param (register WAVEFORM_PICKER_NBSNAPSHOT).

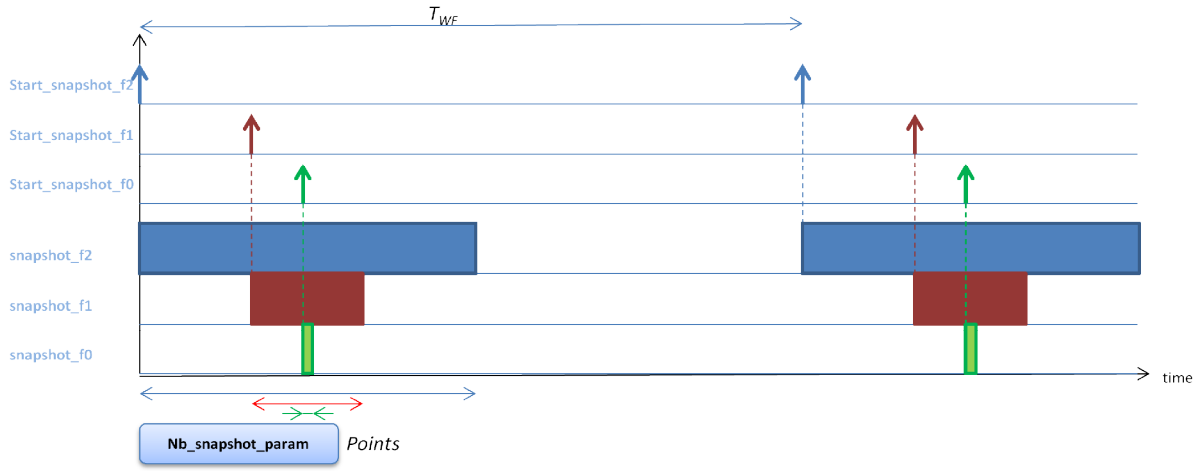


Figure 25 : Numbers of point by snapshot

2.3.6.4.1.2 Buffer Configuration

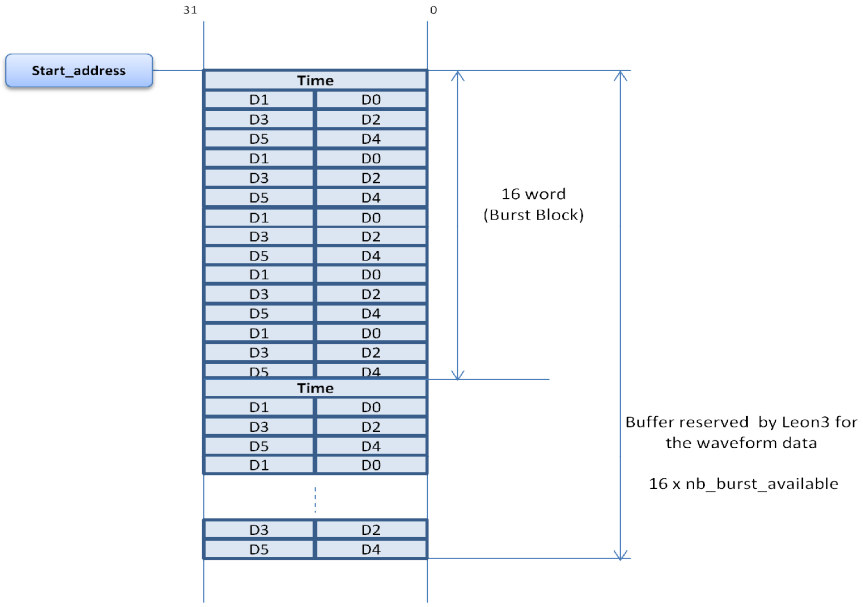


Figure 26 : Waveform Buffer reserved by Leon3

For each channel, the leon3 software must reserve a memory space to receive the waveform data. The Waveform Picker module must be configured to know the beginning and the size of those buffers. This is done by the flight software via the appropriate registers. The first parameter is the start_address (WAVEFORM_PICKER_ADDRESS_Fx), and the second is the number of words which can be written into this buffer (WAVEFORM_PICKER_NBWORDBY BUFFER). Figure 26 shows a buffer example.

2.3.6.4.1.3 Software protocol

The waveform utilization protocol is described in Figure 27. First, the flight software must configure the delta parameters and buffer parameters. After, the flight software configures the burst bits (one for each channel).

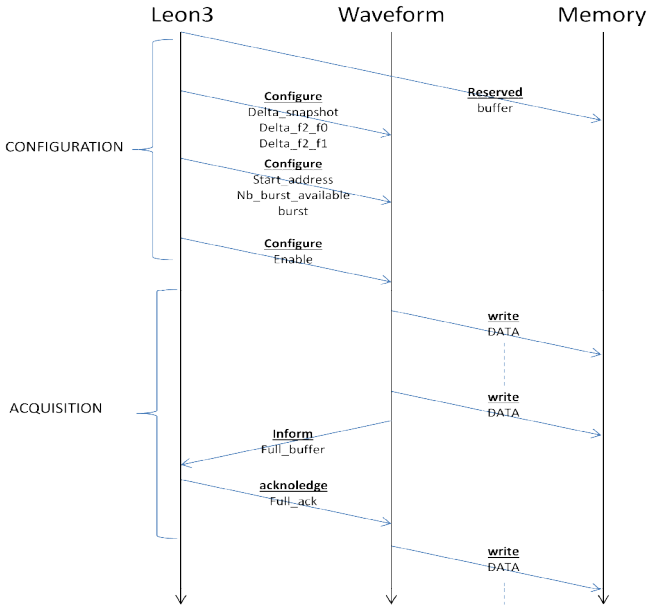


Figure 27 : Protocol Communication between Leon3 and Waveform Picker

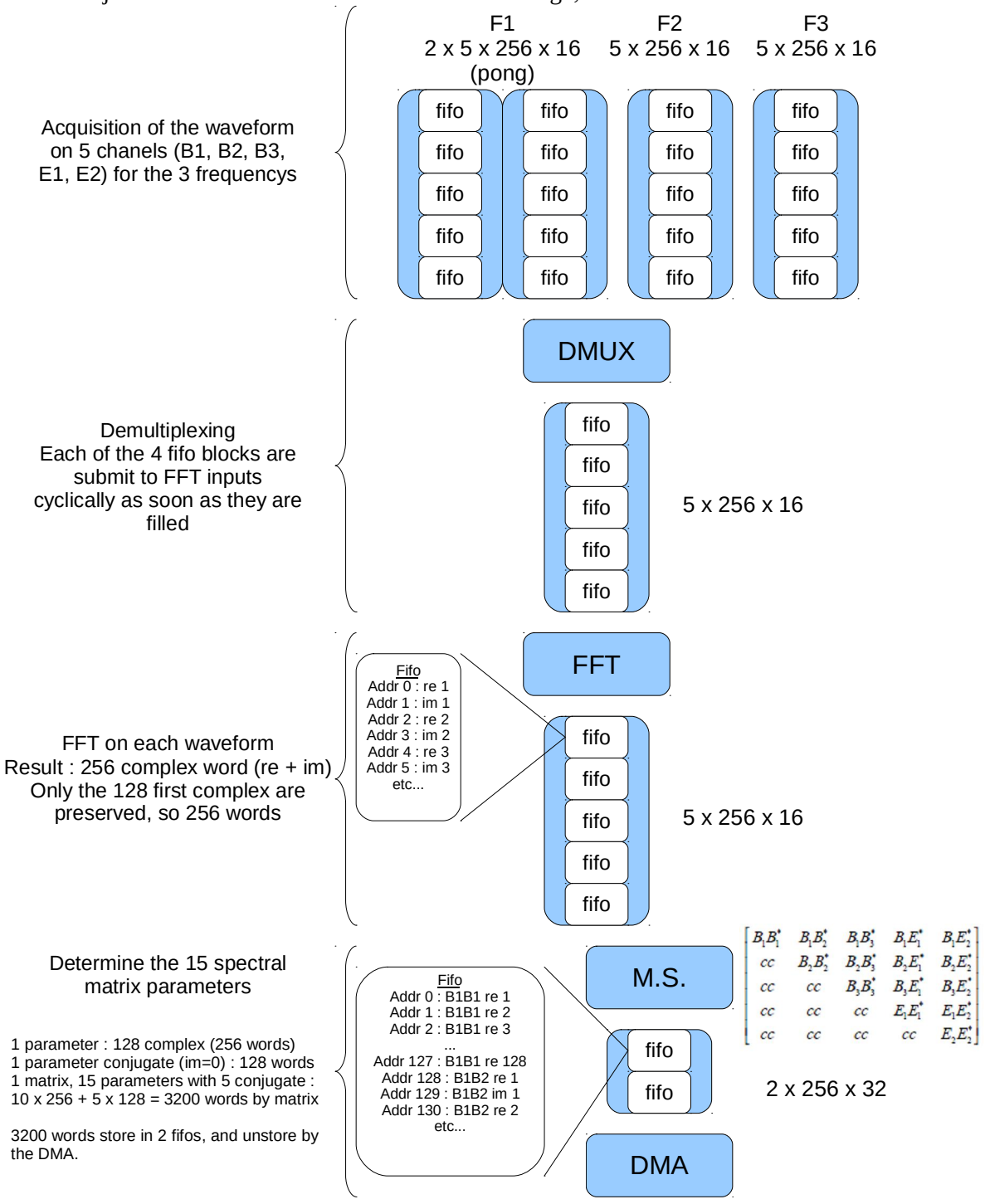
Finally, the flight software can launch the acquisition of data by setting the enable bit and the start date. Once the buffer is full of data, the Waveform Picker module informs the flight software (through

the register full_buffer) that the buffer is full. This is done using an interruption request. As soon as the interruption is raised by the module, the flight software executes the appropriate interrupt service routine thanks to the RTEMS primitives. The flight software can now use this buffer of data for further processing (packet generation and transmission). The flight software can configure a new buffer (a new start address), and send a full_ack (by writing 0 in the full_buffer). Once the full_ack is received, a new acquisition phase can start.

2.3.7 LPP_LFR_SM

Here is a synopsis of the data flow inside LFR, each IP is detailed in the RPW-MEB-LFR-SPC-00062-2-1_LPP_IP_Cores_User_s_Manual documentation.

See the Projet-LeonLFR-A3P3K-Sheldon-DataFlux design, for the vhdl instantiation of this IPs.



TBW

2.3.7.1 Operations

2.3.7.1.1 Software Interface

The table below lists the APB registers in APB LPP_LFR Registers used to control and observe the Spectral Matrix subsystem.

Tableau 5 : Spectral Matrix Registers List

<table>
<tr><th>Address</th><th>Name</th><th>Description</th></tr>
<tr><td>0x80000F00</td><td>SPECTRAL_MATRIX_CONFIG</td><td>Permit to activate the interruption from LFR_SM module and run the Spectral Matrix.
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>31..3</td><td>Not Used</td></tr>
<tr><td>2</td><td>Run the Spectral matrix</td></tr>
<tr><td>1</td><td>Active the interruption when a new Matrix is completely sent in Memory.</td></tr>
<tr><td>0</td><td>Active the interruption when a error occurs.</td></tr>
</table></td></tr>
<tr><td>0x80000F04</td><td>SPECTRAL_MATRIX_STATUS</td><td>Indicates the status of LFR_SM module. A write must be done after a read to ñeraseò the status.
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>31..11</td><td>Not Used</td></tr>
<tr><td>10</td><td>Set when an error ñwrite into Full FIFO_2ò occurs</td></tr>
<tr><td>9</td><td>Set when an error ñwrite into Full FIFO_1ò occurs</td></tr>
<tr><td>8</td><td>Set when an error ñwrite into Full FIFO_0ò occurs</td></tr>
<tr><td>7</td><td>Set when an error ñBuffer Fullò occurs</td></tr>
<tr><td>6</td><td>Set when an error ñBad Componentò occurs</td></tr>
<tr><td>5</td><td>Set when a matrix f2 is ready into buffer 1</td></tr>
<tr><td>4</td><td>Set when a matrix f2 is ready into buffer 0</td></tr>
<tr><td>3</td><td>Set when a matrix f1 is ready into buffer 1</td></tr>
<tr><td>2</td><td>Set when a matrix f1 is ready into buffer 0</td></tr>
<tr><td>1</td><td>Set when a matrix f0 is ready into buffer 1</td></tr>
<tr><td>0</td><td>Set when a matrix f0 is ready into buffer 0</td></tr>
</table></td></tr>
<tr><td>0x80000F08</td><td>SPECTRAL_MATRIX_ADDRESS_F0_0</td><td>Base address of the first buffer where the Spectral Matrix must write the next Spectral Matrix F0</td></tr>
<tr><td>0x80000F0C</td><td>SPECTRAL_MATRIX_ADDRESS_F0_1</td><td>Base address of the second buffer where the Spectral Matrix must write the next Spectral Matrix F0</td></tr>
<tr><td>0x80000F10</td><td>SPECTRAL_MATRIX_ADDRESS_F1_0</td><td>Base address of the first buffer where the Spectral Matrix must write the next Spectral Matrix F1</td></tr>
<tr><td>0x80000F14</td><td>SPECTRAL_MATRIX_ADDRESS_F1_1</td><td>Base address of the second buffer where the Spectral Matrix must write the next Spectral Matrix F1</td></tr>
<tr><td>0x80000F18</td><td>SPECTRAL_MATRIX_ADDRESS_F2_0</td><td>Base address of the first buffer where the Spectral Matrix must write the next Spectral Matrix F2</td></tr>
<tr><td>0x80000F1C</td><td>SPECTRAL_MATRIX_ADDRESS_F2_1</td><td>Base address of the second buffer where the Spectral Matrix must write the next Spectral Matrix F2</td></tr>
</table>

2.3.8 LFR_DMA

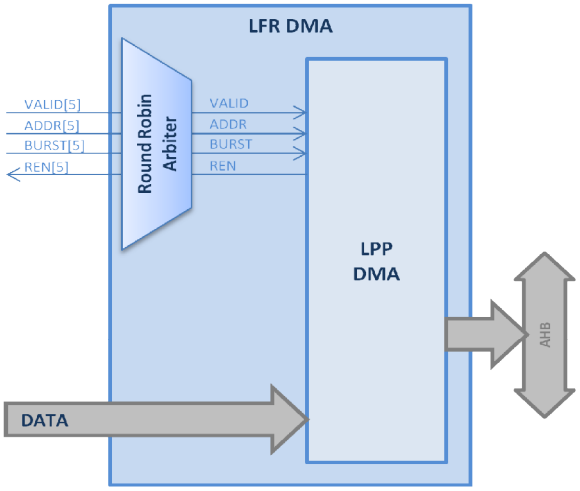


Figure 28 : LFR DMA Block diagram view

2.3.9 LPP_LFR_CAL

2.3.9.1 Block View

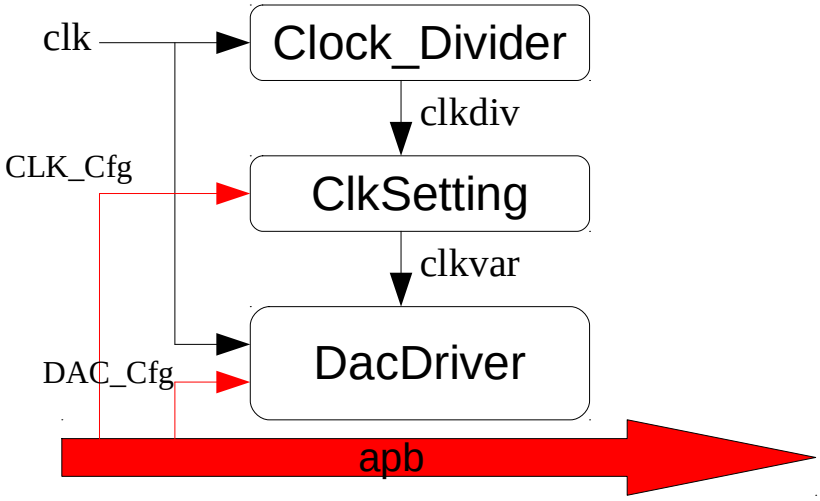


Figure 29 LFR Calibration APB_DAC clock resume and APB register connection

Clk = 24 576 000 Hz = 25 MHz

Clkdiv = clk / 308 = 79 792 Hz = 80 KHz

Clkvar = clkdiv / 2^N =

N	clkvar KHz
0	80
1	40
2	20
3	10
...	...

The purpose of the DacDriver IP, is to deliver the different signals required to drive the DAC, i.e. the serial clock SCLK, the data frame synchronization SYNC, the 16 bits serialized data DATA and a DAC enable signal Cal_EN.

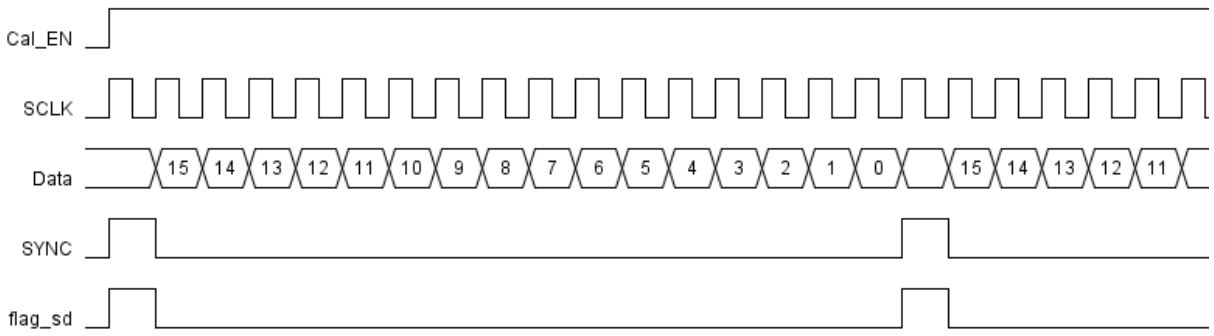


Figure 30 DAC Timing Diagram

2.3.9.2 Generics Parameters

Name	Description	Type	Default
pindex	APB index	integer	0
paddr	APD address	integer	0
pmask	APB Mask address	integer	0xFFF
pirq	output AHB interruption number	integer	0
abits	address size for APD address	integer	8
Nmax	used for the sampling clock (clkvar) division	integer	7

2.3.9.3 Interface

Name	Description	Active	Direction	Size
clk	Clock		Input	1
rst	Reset	Low	Input	1
apbi	APB input interface		Input	
apbo	APB output interface		Output	
Data_IN	Input data		input	16
Cal_EN	Allow the use of the DAC	High	Output	1
Readn	Read fifo for a new input data	low	Output	1
SYNC	Frame synchronization for the data input	High	Output	1
SCLK	Serial Clock		Output	1
DATA	Digital serialized data		Output	1

2.3.9.4 Operations

2.3.9.4.1 Software Interface

2.3.9.4.1.1 Registers

The APB_DAC IP is driven via two registers:

DAC_Cfg, only one bit, DAC_Cfg(0), connect to a enable input, so need to be set to allow the DAC to start.

CLK_Cfg, a three bits register, CLK_Cfg(2 downto 0), contain the N value used by ClkSetting IP for the Sampling clock generation.

2.3.9.4.1.2 Operation

Here is the functioning process step by step for the APB_DAC associated to a fifo:

1. Make sure the DAC is disabled. *DAC_Cfg(0) = 0*
2. Set the sampling clock. *CLK_Cfg(2 downto 0) = N* (ex: 0 for 80 KHz)
3. Fill the FIFO to Full.
4. Set the FIFO to his ReUse state, always full, never empty.
5. And finally enable the DAC. *DAC_Cfg(0) = 1*

A testbench of this procedure is available in the VHD_Lib.

VHD_Lib\LPP_drivers\exemples\BenchDAC_CAL

2.4 SYNTHESIS

The FPGA Flow is run with

Libero Project Manager

Version: 9.1.5.1

Release: v9.1 SP5

The synthesis part is done with:

Synplify Pro

E-201-09A-1

Build 006R

2.4.1 Constraints

2.4.1.1 Pin out

NAME	Pin	Direction	Description
General Purpose			
clk_49_152MHz	D5	IN	Clock input at 49.152MHz
clk100MHz	B3	IN	Clock input at 100MHz
reset	N18	IN	Reset signal active at low value
RAM			
Address [0] [1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15] [16] [17] [18] [19]	 H16 J15 B18 C17 C18 U2 U3 R5 N11 R13 V13 U13 V15 V16 V17 N1 R3 P4 N3 M7	OUT	20 bits of RAM address
Data [0] [1] [2] [3] [4] [5]	 P17 R18 T18 J13 T13 T12	IN/OUT	32 bits of RAM data

[6] [7] [8] [9] [10] [11] [12] [13] [14] [15] [16] [17] [18] [19] [20] [21] [22] [23] [24] [25] [26] [27] [28] [29] [30] [31]	R12 T11 N2 P1 R1 T1 M4 K1 J1 H1 H15 G15 H13 G12 V14 N9 M13 M15 J17 K15 J14 U18 H18 J18 G17 F18		
nSRAM_BE0	U12	OUT	RAM Bank0 Enable (low active)
nSRAM_BE1	K18	OUT	RAM Bank0 Enable (low active)
nSRAM_BE2	K12	OUT	RAM Bank0 Enable (low active)
nSRAM_BE3	F17	OUT	RAM Bank0 Enable (low active)
nSRAM_CE	M6	OUT	RAM Chip Enbale (low active)
nSRAM_OE	N12	OUT	RAM Ouput Enbale (low active)
Space Wire			
Spw1_din	D6	IN	Data In of Space Wire link1
Spw1_sin	C6	IN	Strobe In of Space Wire link1
Spw1_dout	C16	OUT	Data Out of Space Wire link1
Spw1_sout	C4	OUT	Strobe Out of Space Wire link1
Spw2_din	E6	IN	Data In of Space Wire link2
Spw2_sin	C15	IN	Strobe In of Space Wire link2
Spw2_dout	B7	OUT	Data Out of Space Wire link2
Spw2_sout	D7	OUT	Strobe Out of Space Wire link2
ADC			
bias_fail_sw	A3	OUT	Bias/Fail switch control
ADC_OEB_bar_CH [1] [2] [3] [4] [5] [6] [7]	 A14 A10 B10 B13 D13 A11 B12	OUT	Output Enable. One for each ADC (active low)

ADC_smpclk	A15	OUT	Sample Clock for data acquisition
ADC_data [0] [1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13]	 A16 B16 A17 C12 B17 C13 D15 E15 D16 F16 F15 G16 F13 G13	IN	
UARTs			
TAG1	J12	IN	AHB UART Rx
TAG3	L16	OUT	AHB UART Tx
TAG2	K13	IN	APB UART Rx
TAG4	L15	OUT	APB UART Rx
Others			
TAG5	M16	IN/OUT	Unused
TAG6	L13	IN/OUT	Unused
TAG7	P6	IN/OUT	Unused
TAG8	R6	IN/OUT	Unused
TAG9	T4	IN/OUT	Unused
LED [0] [1] [2]	 K17 L18 M17	OUT	LEDs control

2.4.1.2 Clock constraints

Clock Name	Clock Source	Period (ns)	Frequency (MHZ)
clk_49_152MHz	clk_49_152MHz	20.345	49.152
clk100MHz	clk100MHz	10.000	100.000
clk_50	clk_50_s:Q	20.000	50.000
clk_25	clk_25:Q	40.000	25.000
clk_24	clk_24:Q	40.690	24.576

2.4.2 Results

With the MINI-LFR board, a piece of hardware representative of the LFR EM, the last version of LFR (MINI_LFR-WFP_MS-0.1.11) gives this synthesis report:

```
Compile report:
===============

    CORE                        Used:  61643  Total:  75264   (81.90%)
    IO (W/ clocks)              Used:    108  Total:    221   (48.87%)
```

```
    Differential IO             Used:      0  Total:    110   (0.00%)
    GLOBAL (Chip+Quadrant)      Used:      6  Total:     18   (33.33%)
    PLL                         Used:      0  Total:      6   (0.00%)
    RAM/FIFO                    Used:     84  Total:    112   (75.00%)
    Low Static ICC              Used:      0  Total:      1   (0.00%)
    FlashROM                    Used:      0  Total:      1   (0.00%)
    User JTAG                   Used:      0  Total:      1   (0.00%)


Global Information:


    Type            | Used   | Total
    ----------------|--------|--------------
    Chip global     | 5      | 6  (83.33%)*
    Quadrant global | 1      | 12 (8.33%)


    (*) Chip globals may be assigned to Quadrant globals using the Multi-View
Navigator (MVN)
        or Physical Design Constraints (PDC).
        They may also be assigned to Quadrant globals automatically during Layout.


Core Information:


    Type    | Instances    | Core tiles
    --------|--------------|-----------
    COMB    | 49018        | 49018
    SEQ     | 12625        | 12625


I/O Function:


    Type                           | w/o register  | w/ register  | w/ DDR register
    -------------------------------|---------------|--------------|----------------
    Input I/O                      | 22            | 0            | 0
    Output I/O                     | 54            | 0            | 0
    Bidirectional I/O              | 32            | 0            | 0
    Differential Input I/O Pairs   | 0             | 0            | 0
    Differential Output I/O Pairs  | 0             | 0            | 0


I/O Technology:


                                   |   Voltages    |             I/Os
    -------------------------------|-------|-------|-------|--------|-------------
-
    I/O Standard(s)                          | Vcci   | Vref   | Input  | Output  |
```

```
Bidirectional
    --------------------------------|-------|-------|-------|--------|-------------
-
    LVTTL                           | 3.30v | N/A   | 22    | 54     | 32


I/O Placement:


    Locked  : 108 ( 100.00% )
    Placed  :   0
    UnPlaced:   0
```

This version contains the Spectral Matrix and the Waveform Picker modules. The FPGA used on the MINI-LFR and on the EM boards do not have DSP blocks.

2.5 VERIFICATION AND VALIDATION

2.5.1 Verification and Validation

2.5.1.1 Simulation

During the design and for each block, basic functionalities have been simulated with ModelSim. These tests do not cover all cases, but allow the debugging of particular problem.

The testbench files are in the VHD_Lib/designs folder.

The version used for the simulations is:

ModelSim Actel 6.6d

Revision 2010.11

2.5.1.2 Test on MINI-LFR

Once we have a full system debugged in simulation, the next step is to test the design with the MINI-LFR board which is very similar to the LFR Engineering Model (same FPGA, RAM, clocks, with 6ADCs and 2UARTs).

On the MINI-LFR board, the flight software can be executed and extensive validation scripts are run to test the VHDL design. An external connector allows having a look at relevant signals of the VHDL design in order to control the good execution of the VHDL modules.

One shall emphasize that the size of the overall VHDL design is such that it is impossible to build a complete testbench. The use of a representative hardware target is mandatory.

2.5.1.3 Validation on MINI-LFR

According with Software Requirements Specifications (SRS), a set of Test is write in python and can be executed on PC. Those tests interact with Leon3 software on MINI-LFR through a uart link.

Those test launch tasks and loads data. The software and hardware validation are done during this execution.

2.5.1.4 Validation on LFR Engineering Model

MINI-LFR and LFR EM boards are almost the same. The Validation process is redone with the EM-board.

The biggest difference between the 2 boards is that the ADCs are not the same, so the drivers in the VHDL design are different.

APPENDIX A: LIST OF TBC/TBD/TBWS

TBC/TBD/TBW			
Reference/Page/Location	Description	Type	Status