INTERFACE NECTAR ETHERNET

version 1.2

1. Introduction

This note describes the functionalities of the new NECTAR electronics board.
This board is composed of 16 independent electronics channels.
Each channel is composed of one amplifier, a SAM channel, an ADC channel and a FIFO buffer.
The FIFO buffers are embedded in a Altera Cyclone III FPGA.

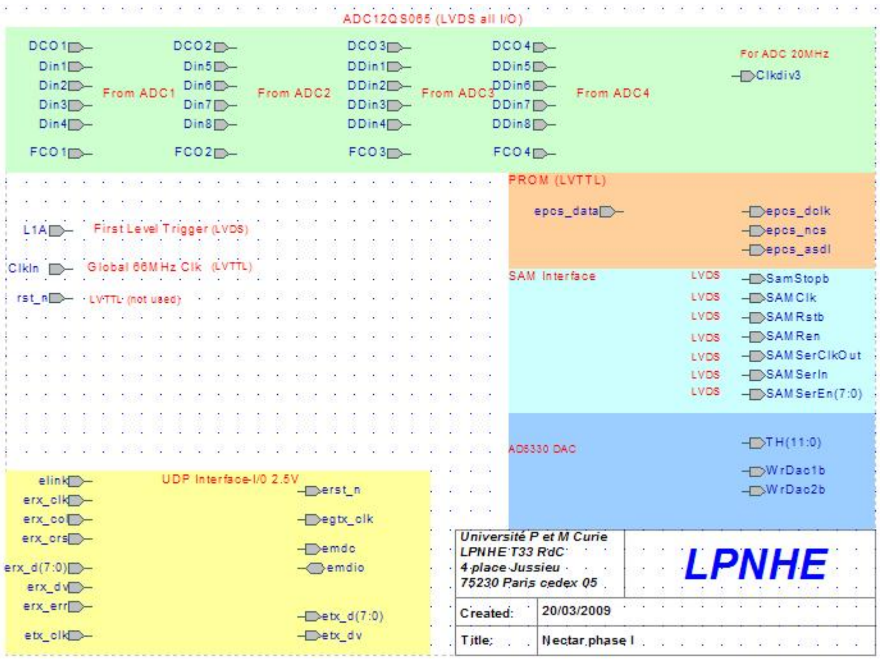


Figure 1.1

2. Software programming

2.1. Register programming

The interface use the following IP Address: 192.168.1.18 and different ports:
port 0x0400, 0x0410, 0x420 and 0x430
The internal register interface is used to define different user internal parameters.
This internal register bank is different from the internal core configuration register.
Do not confuse the two register bank, internal register and internal core configuration register.

2.1.1. Internal register interface

A user 256-register bank is defined.

In order to access the previous register bank, we must send the following UDP frames word1 and word2:

word 1 (32-bit):	[31:24]=Addr	
	[23:16]=Command	
	[15:08]=0xFF	
	[07:00]=0x00	
Command	0x04	Read register
	0x08	Write register

and in the case of a write command

word2 (32-bit): data to be written in the register at Addr.

For example, the EPROM programming used some internal registers (Voir “ § 2.5. page 5 “).

2.1.2. GEDEK core programming

The internal core configuration register is describe below and cannot be accessed directly from Ethernet.

Address	Functionality	Access	Reset Value
0x00	FPGA Board MAC Address (32 LSB)	R/W	Generic Dependent
0x01	FPGA Board IP Address	R/W	Generic Dependent
0x02	Destination MAC Address (32 LSB)	R/W	0x66322E2A
0x03	Destination MAC Address (16 MSB)	R/W	0x00000019
0x04	Destination IP Address	R/W	0xC0A801CF
0x05	Reserved	N/A	N/A
0x06	Reserved	N/A	N/A
0x07	Reserved	N/A	N/A
0x08	Reserved	N/A	N/A
0x09	Reserved	N/A	N/A
0x0A	Virtual Uart Link UDP Port Number (When Available)	R/W	0x00000017
0x0B	Reserved	N/A	N/A
0x0C	Reserved	N/A	N/A
0x0D	Reserved	N/A	N/A
0x0E	Reserved	N/A	N/A
0x0F	Reserved	N/A	N/A

For example the destination IP Address: 0xC0A801CF must be interpreted after hex to dec conversion as C0.A8.01.CF:

C0	-192
A8	-168
01	-01
CF	-207

2.2. PORTS Definition

Different sources of data can be transmitted on UDP. A different Port number is used for each source of data:

DAQPort	x"0400"	for data acquisition
SLCPort	x"0410"	for the board slow control
RegPort	x"0420"	for the internal GEDEK Register bank
NectarPort	x"0430"	for the NeCTAr slow control chip

2.3. Reference design programming

This part of the programming is the user part and gives access to the data.
7 blocks are used to configure the slow control of the board:
All the following words are 32-bit.

CntrlSlc is used to program the threshold for the pixel (Thrshld1 8-bit), the threshold for the pixel sum (Thrshld2 8-bit) and the common mode voltage VMC (12-bit).

```
AAAAAAAA
00007E0C
[0000Thrshld1]
[0000Thrshld2]
[0000VMC]
AAAAAAAA
--
```

CntrlNECTARReg is used to program the internal SAM registers CDR1, CDR2 and TestFCR

```
AAAAAAAA
00007E3E
[0000CDR1]
[0000CDR2]
[000000TestFCR]
[0000CDR6]
[0000CDR7]
AAAAAAAA
--
```

CNTRLNECTARDac is used to program the 32 internal SAM DACs

```
AAAAAAAA
00007E3A
0000MemNum [3:0]
DACL0
DACL1
DACL2
DACL3
DACL4
DACL5
DACL6
DACL7
DACL8
DACL9
DACLA
DACLB
DACLC
DACLD
DACLE
DACLF
AAAAAAAA
--
```

CNTRLNECTARNd is used to define the delay pointer Nd

```
AAAAAAAA
00007E3C
0000YNd              Y=NECTAR number (F=broadcast) Nd[9:0]
AAAAAAAA
--
```

CntrlReadBack

```
AAAAAAAA
00007E40
00000Y00             Y=1 [DAC], Y=2 [Nd], Y=3 [NECTAR CHIP REG]
AAAAAAAA
--
```

CntrlIntReg is used to program and read back the internal GEDEK core registers. In case of read back the message does not include data (AD0-AD4)

When the internal registers are read, the data block is the following:

```
BBBBBBBB
Board MAC Ad.
Board IP Ad.
Dest. MAC Ad. (32 LSB)
Dest. MAC Ad. (16 MSB)
Dest IP Ad.
Status                              bit 0=host detected
BBBBBBBB
```

CntrlIntReg Y=1

```
AAAAAAAA
00007E50
0000000Y             Y=1 write the following registers, Y=0 read back
AD0                  FPGA Board MAC Address (32 LSB)
AD1                  FPGA Board IP Address (32 LSB)
AD2                  Destination MAC Address (32 LSB)
AD3                  Destination MAC Address (16 MSB)
AD4                  Destination IP Address (32 LSB)
AAAAAAAA
--
```

CntrlIntReg Y=0

```
AAAAAAAA
00007E50
0000000Y             Y=0 read back
AAAAAAAA

--
```

CNTRLDAQ is used to define the DAQ parameters

```
AAAAAAAA
00007E30
0000XXXX  [Nf][x][Q/Sampls][T0][TOT] [10][1][1][1][1]
AAAAAAAA
```

The 2 following blocks are used to receive data in either charge mode or sampling mode.

In order to get a correct *IPNum* word, it is mandatory to launch at least one time the **CntrlIntReg Y=0** block to actualize the internal registers.

DAQCHARGE

```
AAAAAAAA
0000EEE0
IPNum
0000Evtcounter (16-bit)
Data block (8 x 32-bit)          [Data1][Data2]
AAAAAAAA
```

DAQSAMPLE

AAAAAAAA
0000EEE3
IPNum
0000Evtcounter (16-bit)
0000Data block (16 x Nf x 16-bit)
AAAAAAAA

2.4. Utility

The utility *ctatest* developed under Linux allows communication with the board *SAMNECTAR*. This utility is able to read a file command text, transform it into binary and send on IP address and on the port desired. This utility can be configured either in client mode (writing on the IP address and port specified) or in server mode (listening on the IP address and port specified). Online help is available by the command: *./ctatest*.

For example: *./ctatest -c 192.168.1.18 -wcta_input_cntrldaq*

The utility takes into account that the given data to transferred is formatted 32 bits, the blocks in the files must not include the 0000 (write AAAA and not 0000AAAA).

2.5. EPROM programming

The design allows to program through ethernet the EPCS FPGA EPROM. This EPROM is used to store the internal FPGA code necessary to operate.

For this purpose, the internal register addresses are used:

x40	EPCSRdRamRdData
x41	EPCSRdRamRdAddress
x42	EpcsStartAddress
x43	reserved for data writing
x44	EPCSStatus

3. NECTAr serial link

In order to program the NECTAr chip, a dedicated serial link is used between the local FPGA and the NECTAr chip.

The serializer inside the FPGA is described in VHDL language. The following lines give the entity of the serializer block.

The communications between the serializer block and the others VHDL blocks are done by a set of two FIFO embedded in the serializer block.

The structure of a FIFO block is the following:

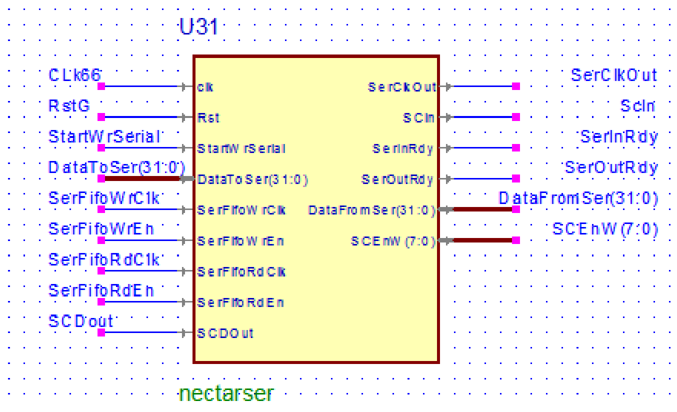
[Command]
[Data block]

The size of the FIFO is defined by the greater block (DAC block) to read/write. For one Nectar chip it needs 16x32-bit and for 8 chips the size must be at least 128x32-bit.

So the data structure must be at least 129x32-bit and the FIFO must be 256x32-bit.

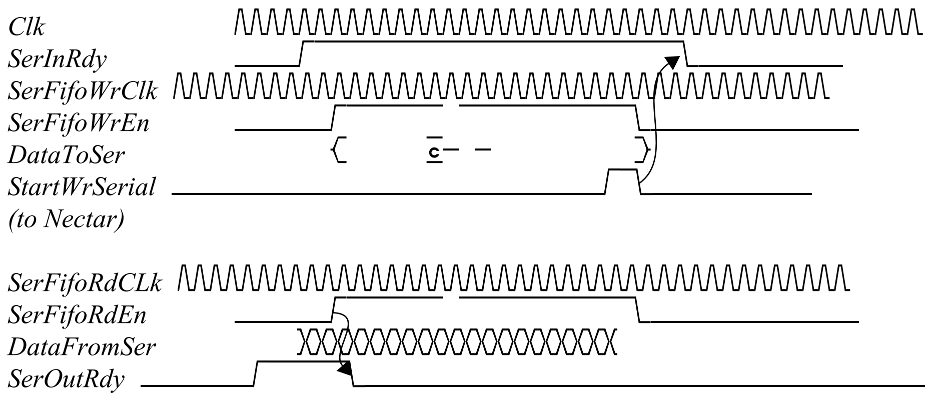
When the FIFO has been written with a data block to serialize, the *StartWrSerial* signal is generated. In other hand, when the output FIFO holds a complete block from the serializer the *SerOutRdy* signal is generated.

The following diagram shows the serializer block and the corresponding VHDL entity.


U31
CLk66
RstG
StartWrSerial
DataToSer(31:0)
SerFifoWrClk
SerFifoWrEn
SerFifoRdClk
SerFifoRdEn
SCDout
clk
Rst
StartWrSerial
DataToSer(31:0)
SerFifoWrClk
SerFifoWrEn
SerFifoRdClk
SerFifoRdEn
SCDOut
SerClkOut
SCIn
SerInRdy
SerOutRdy
DataFromSer(31:0)
SCEnW(7:0)
SerClkOut
SCIn
SerInRdy
SerOutRdy
DataFromSer(31:0)
SCEnW(7:0)
nectarser

```
==============================================================
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity nectarser is
 port(
clk                : in STD_LOGIC;     -- 66MHz
 Rst               : in STD_LOGIC;
 StartWrSerial     : in STD_LOGIC;     -- A data block is ready in the fifo
 DataToSer         : in STD_LOGIC_VECTOR(31 downto 0);-- Fifo input
 SerFifoWrClk      : in STD_LOGIC;      -- For internal input fifo
 SerFifoWrEn       : in STD_LOGIC;      -- Enable for the SerFifoWrClk
 SerFifoRdClk      : in STD_LOGIC;     -- For internal input fifo
 SerFifoRdEn       : in STD_LOGIC;     -- Enable for the SerFifoRdClk
 SCDOut            : in STD_LOGIC;     -- Serial from Nectar
 SerClkOut         : out STD_LOGIC;   -- Clock for Nectar serialiser
 SCin              : out STD_LOGIC;   -- Serial to Nectar
 SerInRdy          : out STD_LOGIC;   -- Serializer readyfor data, fifo empty
 SerOutRdy         : out STD_LOGIC;   -- Serializer ready, all data ready for read
 DataFromSer       : out STD_LOGIC_VECTOR(31 downto 0); -- Fifo output
 SCEnW             : out STD_LOGIC_VECTOR(7 downto 0)    -- Nectar serial enable
);
end nectarser;



architecture nectarser of nectarser is
begin



end nectarser;
==============================================================
```

The following timings shows the signals involved with the remaining FPGA logic.



3.1. The Serial FIFO Command Word

The first word written into the internal FPGA FIFO used before a serial transmission to the NECTAr chip defines how to interpret the following data block.

Block Type	*Value (Hexa)*	*Comment*	*# DATA words*
DAC	*0000001X*	*X=not relevant*	16x32
Nd	*0000002X*	*Y=Chip number* *if Y=0xF then broadcast*	Yy y=[7:0]=Nd
Reg	*0000003X*	*X=not relevant*	CDR1 CDR2 FCR CR6 CR7
	000001XX	DAC Read Back	128x32
	000002XX	Nd Read Back	8x8
	000003XX	Register Read Back	8x5

Tableau 3.1

4. The Serial Peripheral Interface (SPI)

The SPI is a full-duplex, synchronous serial link.

This implementation is based on the Xilinx CoolRunner Serial Peripheral Interface Master.The initial VHDL implementation was done for a 8051 microprocessor interface and has been entirely review and adapted for the Nectar requirements.

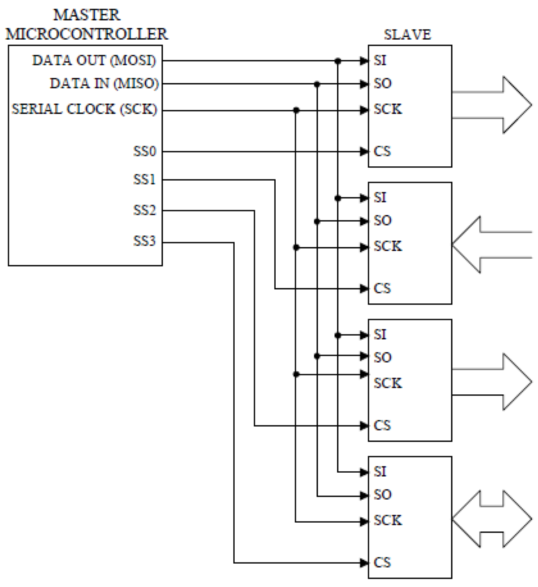


Figure 4.1

4.1. Internal SPI registers

The communication with the SPI are done through 5 internal registers.

Register	Address (Hexa)	
SPISR	0	Status
SPICR	4	Control
SPISSR	8	Slave Select
SPITR	A	Transmit Data
SPIRR	E	Receive Data

Tableau 4.1