MPD FPGA Specifications

Author: Paolo Musico
Paolo.Musico@ge.infn.it

Rev. 3.0
November 30, 2012

Revision History

Rev.	Date	Author	Description
1.0	18/10/10	Paolo Musico	First Draft
1.1	12/11/10	Paolo Musico	Started documenting 2nd release of board
2.0	16/03/11	Paolo Musico	Start better documentation
3.0	11/08/11	Paolo Musico	Documenting 3rd release of FPGA and board

Contents

1

Introduction

The MPD (Multi Purpose Digitizer) Board has been designed to readout 16 APV25 analog data streams.
Here we specify the behaviour of the FPGA (an ALTERA ARRIA GX EP1AGX50DF780C6) mounted on it.

The FPGA will handle:

- VME interface (VME64x with VXS extension).
- Low speed USB interface (very preliminary).
- High speed optical protocol (TBD).
- ADS5281 interfacing (2 x 8 channels, 40 MHz, 12 bit ADC, with DDR serial interface @ 480 Mbit/s).
- I^2C protocol for on-board devices and APV25 configuration.
- APV25 triggering.
- Large memory buffer implemented with external DDR Dram (2 x Micron MT46V64M8: 128 M x 8 bits) (TBD).
- Ethernet 10-100 protocol (TBD).
- Remote reconfiguration (TBD).
- User I/O, LEDs, ...

2

VME Memory Map

The VME bus is the main interface of the board.
The protocol is handled using a previously developed custom block which should implement the following cycles:

- ✓ Single A24 cycles: non privileged and supervisor program and data space, CR/CSR space. They are all used to access configuration ROM and Control/Status registers.
- ✓ Single A32 cycles: non privileged and supervisor program and data space
- ✓ Block transfer A32 cycles: non privileged and supervisor
- ✓ 2eVME cycles: A32 3U and 6U, D32 and D64, master terminated only
- ✓ 2eSST cycles: A32 3U and 6U, D32 and D64, master terminated only

The VME interface decodes various spaces: 512 kB standard CR/CSR and 8 user spaces.

The CR/CSR space must be accessed with A24-D32 cycles. Any A24 single cycle is allowed, not only the dedicated one (AM=0x2F).

The base address of the CR/CSR space is stored in the Base Address Register (BAR), which has a default value, loaded after a RESETb, equal to the geographical address. The base address of the CR/CSR is the following:

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BAR[7:3]					**0**																		

BAR[7:3] = GA[4:0] after a RESETb pulse (or power-up), and can be loaded with any value under software control, thus changing the address mapping of the CR/CSR space.

Using a Vme64x crate the GA[] lines are connected to the corresponding lines on the backplane. In a VME32 crate the GA[] lines can be set using a rotary switch which is present only from the 2nd version of the board.

2.1 Configuration ROM contents

The Configuration Rom contents is the following:

Address	Len	Value	Description

	(bytes)		
0x0000	1		Checksum
0x0004÷0x000C	3	0x001000	Length of ROM to be check-summed
0x0010	1	0x84	CR data access width
0x0014	1	0x84	CSR data access width
0x0018	1	0x02	VME64x CR/CSR space
0x001C	1	0x43	ASCII ‘C’
0x0020	1	0x53	ASCII ‘R’
0x0024÷0x002C	3	0x08_00_30	Manufacturer ID (CERN)
0x0030÷0x003C	4	0x00_03_09_04	Board ID
0x0040÷0x004C	4	0x00_00_00_02	Revision ID
0x005C÷0x0068	4	time_t	Compilation time (MSB first)
0x007C	1	0x01	Program ID code
0x00B0÷0x00B8	3	0x07FBDB	Begin address offset of User CSR area
0x00BC÷0x00C4	3	0x07FBFF	End address offset of User CSR area
0x00E0	1	0x06	Slave characteristics
0x00F4	1	0xFE	Interrupter capabilities
0x0100	1	0x84	Function 0 data access width
0x0104	1	0x84	Function 1 data access width
0x0108	1	0x84	Function 2 data access width
0x010C	1	0x84	Function 3 data access width
0x0110	1	0x84	Function 4 data access width
0x0114	1	0x84	Function 5 data access width
0x0118	1	0x84	Function 6 data access width
0x011C	1	0x84	Function 7 data access width

2.2 User functions

Up to 8 independent user address spaces can be configured.
Each function is identified by a USER_CEb signal, which is activated when the 8 MSB of the 32 bit address loaded into the address counter are matched against the corresponding Adders DEcoder compaRe (ADER) register. Each function can span up to 2^{22} = 4 Mwords of 32 or 64 bit, depending of the transfer size.
The size of the transfer can be only 32 bit and 64 bit. Transfer of 16 bit and 8 bit are not allowed: in this case the module issue a BUS ERROR.
Single 32 bit and block transfer are allowed. Are also allowed 2eVME and 2eSST transfer in 32 or 64 bit mode.

After a RESETb pulse the value of the ADER registers is the following:

ADER0

7	6	5	4	3	2	1	0
GA4	**GA3**	**GA2**	**GA1**	**GA0**	**0**	**0**	**0**

ADER1

7	6	5	4	3	2	1	0

GA4	**GA3**	**GA2**	**GA1**	**GA0**	**0**	**0**	**1**

ADER2

7	6	5	4	3	2	1	0
GA4	**GA3**	**GA2**	**GA1**	**GA0**	**0**	**1**	**0**

ADER3

7	6	5	4	3	2	1	0
GA4	**GA3**	**GA2**	**GA1**	**GA0**	**0**	**1**	**1**

ADER4

7	6	5	4	3	2	1	0
GA4	**GA3**	**GA2**	**GA1**	**GA0**	**1**	**0**	**0**

ADER5

7	6	5	4	3	2	1	0
GA4	**GA3**	**GA2**	**GA1**	**GA0**	**1**	**0**	**1**

ADER6

7	6	5	4	3	2	1	0
GA4	**GA3**	**GA2**	**GA1**	**GA0**	**1**	**1**	**0**

ADER7

7	6	5	4	3	2	1	0
GA4	**GA3**	**GA2**	**GA1**	**GA0**	**1**	**1**	**1**

In the 1st version the user spaces are allocated as the following:

ADER0 NO MORE USED (was pll configuration space)
ADER1 ADS5281 configuration registers
ADER2 I^2C Controller registers
ADER3 Histogrammer registers and memory
ADER4 APV acquisition control registers and FIFOs
ADER5 SDRAM space (not implemented yet)
ADER6 Not used
ADER7 Not used

Given the contents of the ADER registers, the base addresses of the 8 user spaces are the following:

31..24	23..0	
ADER0	**0**	**BASE0**
ADER1	**0**	**BASE1**
ADER2	**0**	**BASE2**
ADER3	**0**	**BASE3**

ADER4	**0**	**BASE4**
ADER5	**0**	**BASE5**
ADER6	**0**	**BASE6**
ADER7	**0**	**BASE7**

2.2.1 ADS5281 configuration registers

31	30	29..24	23..0	
Start ADC1	**Start ADC0**	**N.U.**	**Data**	**BASE1 (W)**

Writing at ADER1 address bit 23..0 is the value to be loaded into the selected ADS5281. Bit 30 start the serialization into ADC0 (U12) related to lower 8 channels, bit 31 start the serialization into ADC1 (U31) related to upper 8 channels.

31	30	29..24	23..0	
Status ADC1	**Status ADC0**	**N.U.**	**Data Readback**	**BASE1 (R)**

Reading at ADER1 address bit 23..0 is the value previously loaded. Bit 30 and 31 reports the status of the serialization process: 1 = Serialization in progress, 0 = Done,

2.2.2 I^2C controller registers

31..8	7..0	
N.U.	**Clock Prescaler low**	**BASE2 + 0x00 (RW)**
N.U.	**Clock Prescaler high**	**BASE2 + 0x04 (RW)**
N.U.	**Control Register**	**BASE2 + 0x08 (RW)**
N.U.	**Transmit Register**	**BASE2 + 0x0C (W)**
N.U.	**Receive Register**	**BASE2 + 0x0C (R)**
N.U.	**Command Register**	**BASE2 + 0x10 (W)**
N.U.	**Status Register**	**BASE2 + 0x10 (R)**
N.U.	**ApvReset[0]**	**BASE2 + 0x1C (RW)**

For detailed informations see the I^2C - Master Core Specification v 0.9.
ApvReset[0] is connected to APV RESETb HW line (NEGATED)

2.2.3 Histogrammer registers and memory

For modularity histogramming of 16 channels has been divided in 2 independent blocks.
The first block handles channels from 0 to 7 and the second one handles channels from 8 to 15.

To make an histogram:

- clear the memory (write 0 to all the 4096 words);
- set histogramming channel and **Process** bit in the control register;
- wait for a given time;
- clear the **Process** bit in the control register;
- read the memory.

First block histogramming memory:

31..0
Histo Data ch 0..7

BASE3 .. (BASE3 + 0x3FFC) (RW)

First block control and status registers:

31..8	7	6..3	2..0
N.U.	**Process**	**N.U.**	**Channel select 0..7**

BASE3 + 0x4000 (W)

31	30..16	15..0
Running	**N.U.**	**Control Register Read Back**

BASE3 + 0x4000 (R)

First block measurements counter:

31..0
Histo Count

BASE3 + 0x4004 (R)

Second block histogramming memory:

31..0
Histo Data ch 8..15

(BASE3 + 0x8000)..(BASE3 + 0xBFFC) (RW)

Second block control and status registers:

31..8	7	6..3	2..0
N.U.	**Process**	**N.U.**	**Channel select 8..15**

BASE3 + 0xC000 (W)

31	30..16	15..0
Running	**N.U.**	**Control Register Read Back**

BASE3 + 0xC000 (R)

Second block measurements counter:

31..0
Histo Count

BASE3 + 0xC004 (R)

2.2.4 APV acquisition control registers and FIFOs

APV channels data FIFO and related word counter:

31..16	15..0	
0	**APV Data ch 0**	**(BASE4 + 0x0000) (R)**
0	**APV Data ch 1**	**(BASE4 + 0x4000) (R)**
0	**APV Data ch 2**	**(BASE4 + 0x8000) (R)**
0	**APV Data ch 3**	**(BASE4 + 0xC000) (R)**
0	**APV Data ch 4**	**(BASE4 + 0x10000) (R)**
0	**APV Data ch 5**	**(BASE4 + 0x14000) (R)**
0	**APV Data ch 6**	**(BASE4 + 0x18000) (R)**
0	**APV Data ch 7**	**(BASE4 + 0x1C000) (R)**
0	**APV Data ch 8**	**(BASE4 + 0x20000) (R)**
0	**APV Data ch 9**	**(BASE4 + 0x24000) (R)**
0	**APV Data ch 10**	**(BASE4 + 0x28000) (R)**
0	**APV Data ch 11**	**(BASE4 + 0x2C000) (R)**
0	**APV Data ch 12**	**(BASE4 + 0x30000) (R)**
0	**APV Data ch 13**	**(BASE4 + 0x34000) (R)**
0	**APV Data ch 14**	**(BASE4 + 0x38000) (R)**
0	**APV Data ch 15**	**(BASE4 + 0x3C000) (R)**
Used Word ch 1	**Used Word ch 0**	**(BASE4 + 0x80000) (R)**
Used Word ch 3	**Used Word ch 2**	**(BASE4 + 0x80004) (R)**
Used Word ch 5	**Used Word ch 4**	**(BASE4 + 0x80008) (R)**
Used Word ch 7	**Used Word ch 6**	**(BASE4 + 0x8000C) (R)**
Used Word ch 9	**Used Word ch 8**	**(BASE4 + 0x80010) (R)**
Used Word ch 11	**Used Word ch 10**	**(BASE4 + 0x80014) (R)**
Used Word ch 13	**Used Word ch 12**	**(BASE4 + 0x80018) (R)**
Used Word ch 15	**Used Word ch 14**	**(BASE4 + 0x8001C) (R)**

All **APV Data ch x** FIFO can be accessed from the given address for 4 K words, to permit block transfers with auto increment on address lines.
MSB of the Used Word registers is the corresponding FIFO Full flag, thus avoiding that if channel FIFO is full the corresponding Used Word is 0. In this release the FIFO size is 2048 words
Space from 0x40000 to 0x7FFFF has been reserved for possible future expansion.

If DAQ MODE is 0x1, the readout data come form the APV frame decoder.
If DAQ MODE is 0x3, the readout data come form the baseline subtractor and threshold cutter.
Used Words and FIFO flags behave in the same way.

If Event Building is enabled all the data must be read from Channel 0 addresses (Data FIFO, Used words and flags).

Apv Readout Control and Status Registers:

31..16	15..0	
FIFO FULL Flag	**FIFO EMPTY Flag**	**BASE4 + 0x80020 (R)**
SYNCED	**ERROR**	**BASE4 + 0x80024 (R)**

FIFO EMPTY Flag & FIFO Full Flag = corresponding FIFO flag on READ side
ERROR = FIFO Full Flag on WRITE side
SYNCED = channel has sync pulses present

31..0	
MISSED TRIGGER	**BASE4 + 0x80028 (R)**
READOUT CONFIG	**BASE4 + 0x80030 (RW)**
TRIG GEN CONFIG	**BASE4 + 0x80034 (RW)**

MISSED TRIGGER = n. of trigger pulses received but not sent to APVs, due to input FIFO congestion

READOUT CONFIG:

- [2:0] = DAQ MODE: 0 = disabled; 1 = APV frame decoding; 2 = fill up FIFO once with ADC samples; 3 = processed mode (baseline and pedestal subtraction, threshold cut); 4..7 = TBD
- [15] = TEST MODE: Not used in this release
- [27:16] = COMMON OFFSET to be added to all APV data
- [28] = Enable Baseline Subtraction
- [30] = Enable Event Building
- [31] = All FIFO Clear

TRIG GEN CONFIG:

- [7:0] = RESET LATENCY: issue trigger pulses only after this latency once enabled
- [11:8] = MAX TRIG OUT: number of trig pulses that have to be sent to APVs every incoming pulse, every 3 clock cycles (minimum spaced)
- [14:12] = TRIG MODE: 0 = disabled; 1 = generates APV trigger every input rising edge; 2 = generates MAX TRIG OUT APV trigger (100) every input rising edge; 3 = generates a Calibration command (110) followed by MAX

TRIG OUT Trigger (100) pulse after given latency (see bit 31..24); 4..7 = TBD

- [15] = TEST MODE: Not used in this release
- [16] = Enable SYNC from P0 VXS connector
- [17] = Enable SYNC from front panel INPUT[1] line
- [18] SOFTWARE TRIGGER: a 100 pulse is generated to APVs every rising edge of this bit
- [19] SOFTWARE CLEAR: a 101 pulse is generated to APVs every rising edge of this bit
- [20] = Not Used
- [21] = Enable TRIG1 line on P0 VXS connector as trigger
- [22] = Enable TRIG2 line on P0 VXS connector as trigger
- [23] = Enable front panel INPUT[0] line as trigger
- [31:24] = Latency (in clock cycles) between calibration command and successive trigger pulse. Effective latency is tis number + 4.

31..28	27..16	15..12	11..0	
N.U.	**ONE Thr**	**N.U.**	**ZERO Thr**	**BASE4 + 0x80038 (RW)**

ZERO Thr = a value below this is considered as logic ‘0’
ONE Thr = a value above this is considered as logic ‘1’

31..24	23..16	15..0	
N.U.	**SYNC Period**	**ENABLE**	**BASE4 + 0x8003C (RW)**

ENABLE = enable mask for all channels: Enable[0] = 1 enables channel 0 and so on
SYNC Period = n. of clock cycles between 2 APV sync pulses. Used only for checking SYNC pulse presence while there are no APV data frames (i.e. no trigger are sent out).

Pedestals and Thresholds RAMs

Each RAM contains 128 12-bit pedestal values to be subtracted from the incoming APV data or threshold values to be compared with processed data.
Before pedestal subtraction the COMMON OFFSET value is added to avoid negative numbers.
These memories are dual port RAM shared with the processor FSMs. To read and write from these RAMs the corresponding channel must be disabled (ENABLE register bit cleared).
If Pedestal value is 0xFFF the corresponding APV channel is not used in baseline computation.
If Threshold value is 0xFFF the corresponding APV channel is always discarded from the output data set.

31..16	15..0	
Pedestal APV 1	**Pedestal APV 0**	**(BASE4 + 0x40000) ÷ (BASE4 + 0x401FC) (RW)**
Pedestal APV 3	**Pedestal APV 2**	**(BASE4 + 0x44000) ÷ (BASE4 + 0x441FC) (RW)**
Pedestal APV 5	**Pedestal APV 4**	**(BASE4 + 0x48000) ÷ (BASE4 + 0x481FC) (RW)**
Pedestal APV 7	**Pedestal APV 6**	**(BASE4 + 0x4C000) ÷ (BASE4 + 0x4C1FC) (RW)**
Pedestal APV 9	**Pedestal APV 8**	**(BASE4 + 0x50000) ÷ (BASE4 + 0x501FC) (RW)**
Pedestal APV 11	**Pedestal APV 10**	**(BASE4 + 0x54000) ÷ (BASE4 + 0x541FC) (RW)**
Pedestal APV 13	**Pedestal APV 12**	**(BASE4 + 0x58000) ÷ (BASE4 + 0x581FC) (RW)**
Pedestal APV 15	**Pedestal APV 14**	**(BASE4 + 0x5C000) ÷ (BASE4 + 0x5C1FC) (RW)**

31..16	15..0	
Threshold APV 1	**Threshold APV 0**	**(BASE4 + 0x60000) ÷ (BASE4 + 0x601FC) (RW)**
Threshold APV 3	**Threshold APV 2**	**(BASE4 + 0x64000) ÷ (BASE4 + 0x641FC) (RW)**
Threshold APV 5	**Threshold APV 4**	**(BASE4 + 0x68000) ÷ (BASE4 + 0x681FC) (RW)**
Threshold APV 7	**Threshold APV 6**	**(BASE4 + 0x6C000) ÷ (BASE4 + 0x6C1FC) (RW)**
Threshold APV 9	**Threshold APV 8**	**(BASE4 + 0x70000) ÷ (BASE4 + 0x701FC) (RW)**
Threshold APV 11	**Threshold APV 10**	**(BASE4 + 0x74000) ÷ (BASE4 + 0x741FC) (RW)**
Threshold APV 13	**Threshold APV 12**	**(BASE4 + 0x78000) ÷ (BASE4 + 0x781FC) (RW)**
Threshold APV 15	**Threshold APV 14**	**(BASE4 + 0x7C000) ÷ (BASE4 + 0x7C1FC) (RW)**

5

Operation

TBD

6

Implementation

6.1 TRIGGER GENERATOR MODULE

The Trigger Generator Module (TGM) generates pulses on the APV Trigger line.
It has the following inputs:

- Trigger and Fast Reset signals from front panel LEMO connectors or backplane
- Software Trigger and Fast Reset bits form Configuration Register
- Trigger Mode bits form Configuration Register
- Reset Latency value form Configuration Register
- Number of consecutive trigger form Configuration Register
- Signals form data FIFOs indicating possible congestions

and the following outputs:

- APV Trigger line
- Number of missed trigger (received, but not sent to APVs)
- Busy line

The behaviour can be summarized as follows.

- APV Trigger = ‘110’ every rising edge of Calibration signal
- APV Trigger = ‘101’ every rising edge of (Fast Reset signal or Software Fast Reset bit) or Mode bits changing from ‘000’ to something different
- APV Trigger = ‘100’ in 2 cases:
    - Mode bits = ‘001’: Space Available in all FIFOs and rising edge of (Trigger signal or Software Trigger bit) and latency counter >= Reset Latency
    - Mode bits = ‘010’: generates Number of consecutive trigger starting every rising edge of (Trigger signal or Software Trigger bit)
- The Number of missed trigger is cleared every ‘101’ APV trigger line sequence and incremented if a trigger pulse can be handled but there is no space in the FIFO to store data
- The Busy line is activated if there is no space available in the data FIFO and the Mode bits are different from ‘000’

6.2 DATA ACQUISITION MODULE

It is composed by 16 identical submodules, arranged in groups of 8 for modularity with the ADS5281.
Each channels has the following inputs:

- Enable bit from Control Register
- ADC output data stream
- Mode bits from configuration register
- FIFO Clear bit from configuration register
- TEST Mode bit from configuration register
- Thresholds to identify logic ‘1’ and logic ‘0’ within the serial stream
- FIFO read signal

and the following outputs:

- Output data bus form the FIFO
- FIFO flags (empty, full) and number of used words
- FIFO almost full (an additional event will overrun the buffer)

The behaviour can be summarized as follows.

7 References

- ANSI/VITA 1-1994 “American National Standard for VME64”
- ANSI/VITA 1.1-1997 “American National Standard for VME64 Extensions”
- ANSI/VITA 23-1998 “American National Standard for VME64 Extensions for Physics and Other Applications”
- VITA 1.5-1999 “Draft Standard for Trial Use Approved bi the VITA Standards Organization for the 2eSST”
- ALTERA “Arria GX Arria GX Device Handbook, Volume 1 & 2”
- Texas Instruments, ADS5281 Datasheet
- OpenCores, “I^2C – Master Core Specification”, July 3, 2003, Rev 0.9