Global Muon Trigger Module

9U-Version

H. Bergauer, K. Kastner, T. Nöbauer, I. Mikulec, M. Padrta,
H. Sakulin, A. Taurok

Dez 08: SORTER description added

1 Description

1.1 Versions history

HB 2014-04-30:
ROP => v1006
SORT => v1003
INx => v1008
Description: Added one SRL16 for latency delay with fixed value of 16, to get greater delays in modules BXReadCounter_IN_SRT.vhd and BXReadCounter.vhd (for ROP), because L1A from TCDS has more latency (than from TCS).

2 Interfaces

2.1 Regional Trigger data

2.2 Output to Global Trigger

3 Common logic circuits

3.1 Configuration of FPGAs and Proms

Standard configuration modes:
1) Normally all FPGA chips are configured at Power-Up by the content of the PROMs. Therefore all FPGAs are soldered to run in MASTER SERIAL mode during configuration from Proms. *(Solder SMD-resistors on the MEZZ896 board to set M2=M1=M0=LOW ➔ MASTER MODE).*
2) New permanent versions are loaded into the PROMs by VME-JTAG with only the PROMs included into the JTAG-chain. *This method cannot configure the VME64 chip.*
3) The Altera Byte Blaster has to be used to load a new permanent version for the VME64 chip.
Optional configuration modes for tests:
1) To load a new temporary version for tests the FPGA chips (INB, INC, IND, INF, LFB, LFF, AUB, AUF, SRT) can be reconfigured directly via VME-JTAG but then the Configuration MODE resistors have to be re-soldered before to M2=1, M1=0, M0=1.
2) To configure the FPGA chips directly by VME the Configuration MODE resistors have to be re-soldered before to M2=1, M1=1, M0=1. This method also cannot configure the ROP and VME64 chip.

Configuration Mode	M2	M1	M0	CCLK Direction	Data Width	Dout
MasterSerial	**0**	**0**	**0**	**out**	**1**	**yes**
Slave Serial	1	1	1	in	1	yes
Master SelectMAP	0	1	1	out	8	no
Slave SelectMAP	1	1	0	in	8	no
Boundary Scan	**1**	**0**	**1**	**N/A**	**1**	**no**

Table 1 Configuration Modes for Virtex2 FPGAs

3.1.1 Default Configuration of FPGAs from PROMs

The FPGAs are set to **MASTER SERIAL** mode.

3.1.1.1 VME64 chip

- POWER UP — At power-up the Chip is configured automatically from PROM

3.1.1.2 ROP chip

- POWER UP — At power-up the Chip is configured automatically from PROM
- NSYSRES — crate reset

3.1.1.3 INx, LFx, AUx, SRT chips

- POWER UP — At power-up the Chip is configured automatically from PROM
- V_NPROG — VME command start a reconfiguration
- NSYSRES — crate reset

3.1.2 New permanent configuration file ➔ PROMs

New permanent configuration files are loaded into PROMs using JTAG.

3.1.2.1 VME64 Prom

- Byte Blaster ➔ JTAG CHAIN_A
- Backplane-JTAG (altera) ➔ JTAG CHAIN_A (=future option)

3.1.2.2 INx, LFx, AUx, SRT and ROP Proms

****** Exclude the FPGA chips from the JTAG chain!! ******

- Parallel Cable IV ➔ JTAG CHAIN_X
- VME-JTAG from ROP ➔ JTAG CHAIN_X
    - ROP-FPGA is skipped by jumpers (default position).
    - Other FPGAs might also be skipped by jumpers.
    - After reconfiguration from Proms (power-up, NSYSREs) the former design is lost.
    - ROP is used to change it's own logic circuits.
- Backplane-JTAG (xilinx) ➔ JTAG CHAIN_X (=future option)

3.1.3 Alternative FPGA configuration methods

Alternative FPGA configuration methods might be used during hardware tests. The configuration mode SMD-jumpers have to be soldered accordingly.

3.1.3.1 VME64

Set VME64 chip to JTAG mode.

- Byte Blaster ➔ JTAG CHAIN_A

3.1.3.2 ROP

Set ROP chip to JTAG mode.

- Parallel Cable IV ➔ JTAG CHAIN_X with ROP chip included.

3.1.3.3 INx, LFx, AUx, SRT

Set FPGAs to JTAG mode:

- Parallel Cable IV ➔ JTAG CHAIN_X
- VME-JTAG from ROP ➔ JTAG CHAIN_X with ROP chip skipped to avoid a crash.

Set FPGAs to SLAVE SERIAL mode:

- VME with CONFIG-Signals (CCLK, NPROG, DIN…) from ROP chip.
    - Set INx, LFx, Aux, SRT before to **SLAVE** mode by soldering on the MEZZ896 board the SMD-RESISTORS M2=M1=M0=HIGH.

3.1.4 JTAG Chains on GMT board

CHAIN_A: VME64x and its Proms <== ByteBlaster or Backplane-JTAG
CHAIN_X: INx, LFx, , AUx, SRT, ROP and their PROMS are controlled by
<== VME_JTAG, ParallelCableIV or Backplane-JTAG

The ROP chip will be skipped to keep it running during VME_JTAG configuration.

3.2 RESET concept

The following points show all reset options for the GMT in the Global Trigger crate.

3.2.1 POWER OFF and ON

To switch the GT-crate off is the last option to reset non-working Global Trigger electronics.

3.2.2 NSYSRES ➔ configuration of FPGAs

The common crate reset signal NSYSRES starts the configuration procedure for all FPGAs except the VME64 chip. It pulls the NPROG net to a low voltage level forcing each FPGA (master) to reconfigure from Proms.

3.2.3 RESET_DCM_xxx

ROP sends 9 signals **RESET_DCM_xxx** to the FPGAs (INx, LFx, Aux, SRT) resetting the DCM units and therefore re-synchronising the chips to the board CLK.

The DCM unit of ROP cannot be reset. In case of problems the ROP chip has to be reconfigured by NSYSRES = crate reset.

Net name	From ROP pins	To xxx pins
RESET_DCM_AUF	F16	AUF: AJ19

RESET_DCM_LFF	C16	LFF: AK17
RESET_DCM_LFB	**D17 ok**	LFB: AH18
RESET_DCM_AUB	**J16 ok**	AUB: AJ19
RESET_DCM_SRT	F17	SRT: AH19
RESET_DCM_INB	**J17 ok**	INB: AG17
RESET_DCM_IND	**G17 ok**	IND: AG17
RESET_DCM_INC	**H17 ok**	INC: AG17
RESET_DCM_INF	C15	INF: AG17

3.2.4 RESET_xxx and INACTIVE ➔STARTUP

The ROP chip sends 9 RESET_xxx signals to the FPGAs (INx, LFx, AUx, SRT) to reset the STARTUP modules inside the chips and the common INACTIVE signal enables the IO-pins to switch from high-Z to active mode.

RESET_xx ➔ GSR pin of STARTUP
INACTIVE ➔ GTS pin of STARTUP

The RESET_xx should reload the initial default values into all registers.

3.2.5 L1RES, BCRES, L1A ➔ reset State Machines and Counters

The Trigger Control System sends via the backplane the signals L1RES, BCRES, L1A and Event Counter Reset to reset state machines and counters.
The AUB and AUF chip do not receive the BCRES signal directly from the Backplane.
The ROP chip generates a 'BCRES' signal generated either by a VME command or by the synchronous BCRES signal broadcasted by the TCS board via the TIM board.
The AUF and AUB chip receive a BCRES signal only via the 'Dummy_AUB(F)' lines.
Remark: In the ROP chip there is no INFF and no OUTFF for the BCRES signal!! ← TO BE REPAIRED !!!

3.2.6 VME commands ➔ reset State Machines and Counters

Also VME commands can be used to reset logic circuits inside the FPGAs.

3.3 Configuration at Power-UP and by SYSREST*

Power-Up respectively NPROG=low keep clearing configuration memory of the chip. After 2 memoring clearing cycles. The chip waits until NINIT has been released and becomes high. Then the Master serial CCLK begins loading the configuration data into the FPGA. If the CRC check finds an error afterwards the NINIT will be pulled low and startup aborted. If CRC check is ok the STARTUP sequence switches the chip into the operational mode.

3.4 Status monitoring

The INx, LFx, AUx, SRT chips send their **DCM_LOCKED** status signals and **2 STATUS bits** to the ROP chip. The ROP chip combines all status signals to a common 4 bit status code and sends the error code if a FPGA has lost synchronisation to the 40 MHz clock.
The 4 bit STATUS bits go via the backplane to the FDL board. On the FDL board the states of all boards are combined and the result is sent as the GT-crate status to the central trigger control board TCS.
The ROP writes the DCM locked signals and the STATUS bits into a status register that can be accessed by VME software.
A front panel LED shows also the combined status of the DCM_LOCKED signals.

3.4.1 Combining status signals

Each GMT chip sends 2 coded status bits to the ROP chip according to the table below.

Code	Status with examples
00	All ok
01	Warning buffer overflow (75% full derandomizing buffer)
10	Out_of sync: BC-cntr error, Derand-Buffer not empty at same time….etc
11	Error or FPGA is not configured (Pull-up resistors provide ‘11’.)

First the 2 bit codes are decoded back to singles state bits. The warning, out_of_sync and error states of all chips are ‘OR’ed. The inverted DCM_LOCKED_xx signals are also included as a error bits. The ROP chip provides the BUSY and READY state as set by the software. Then the results are coded again into a 4-bit code to be sent to the FDL board.

The following states can be formed with the following 4-bit code:

0001 WARNING // = OR of all warnings from INx, SRT and ROP chip
0010 OUT_OF_SYNC // =OR of all GMT chips
0100 BUSY // = set by VME software (CMD_REG in ROP chip)
1000 READY // = set by VME software (CMD_REG in ROP chip)
1100 ERROR // = OR of all GMT chips

A priority logic sends only the status with highest rank as a 4 bit code to the FDL board.
Highest priority: ERROR
2^{nd} rank: OUT_OF_SYNC
3^{rd} rank: BUSY
4^{th} rank: WARNING
Lowest rank: READY
The Global Muon Trigger does not use other codes, which would be interpreted on the FDL board as ‘BAD CODE’. If the ROP chip is not configured or if the GMT board is not inserted then the FDL board receives either ‘0000’ or ‘1111’ as the ‘DISCONNECTED’ status.

4 VME Addresses Overview

4.1 Address Table

```
Status of  1.June 2006  FONT for Table:  COURIER 11 pts
*********************************************************************
* Global Muon Trigger address table generated by
extract_address_table.pl on Tue Feb  1 17:29:30 MET 2005
*********************************************************************
*  key                    space   map   address        mask   read write
description
*********************************************************************
*
* addresses extracted from ROPChip/src/ROPVMEAddrMap.vhd
*
ROP_base                        memory  0   00000000 0000ffff   1   1
INF_base                        memory  0   00100000 0000ffff   1   1
INC_base                        memory  0   00200000 0000ffff   1   1
IND_base                        memory  0   00300000 0000ffff   1   1
INB_base                        memory  0   00400000 0000ffff   1   1
AUF_base                        memory  0   00500000 0000ffff   1   1
LFF_base                        memory  0   00600000 0000ffff   1   1
LFB_base                        memory  0   00700000 0000ffff   1   1
AUB_base                        memory  0   00800000 0000ffff   1   1
SRT_base                        memory  0   00900000 0000ffff   1   1
ALL_base                        memory  0   00A00000 0000ffff   1   1
MEM_end                         memory  0   009FFFFE 0000ffff   1   1
ROP_chip_id0_raddr              memory  0   00000000 0000ffff   1   0
ROP_chip_id1_raddr              memory  0   00000002 0000ffff   1   0
ROP_chip_rev0_raddr             memory  0   00000004 0000ffff   1   0
ROP_chip_rev1_raddr             memory  0   00000006 0000ffff   1   0
ROP_JTAG_base                   memory  0   00000020 0000ffff   1   1
ROP_reset_addr                  memory  0   00000030 0000ffff   1   1
ROP_reset_dcm_addr              memory  0   00000032 0000ffff   1   1
ROP_command_addr                memory  0   00000034 0000ffff   1   1
ROP_JTAG_enable_addr            memory  0   00000036 0000ffff   1   1
ROP_prog_enable_addr            memory  0   00000040 0000ffff   1   1
ROP_nprog_addr                  memory  0   00000042 0000ffff   1   1
ROP_init_cmd_addr               memory  0   00000044 0000ffff   1   1
ROP_init_status_raddr           memory  0   00000046 0000ffff   1   0
ROP_done_raddr                  memory  0   00000048 0000ffff   1   0
ROP_din_INF_addr                memory  0   0000004A 0000ffff   1   1
ROP_din_INC_addr                memory  0   0000004C 0000ffff   1   1
ROP_din_IND_addr                memory  0   0000004E 0000ffff   1   1
ROP_din_INB_addr                memory  0   00000050 0000ffff   1   1
ROP_din_AUF_addr                memory  0   00000052 0000ffff   1   1
ROP_din_LFF_addr                memory  0   00000054 0000ffff   1   1
ROP_din_LFB_addr                memory  0   00000056 0000ffff   1   1
ROP_din_AUB_addr                memory  0   00000058 0000ffff   1   1
ROP_din_SRT_addr                memory  0   0000005A 0000ffff   1   1
ROP_dcm_locked_raddr            memory  0   00000060 0000ffff   1   0
ROP_status0_raddr               memory  0   00000062 0000ffff   1   0
ROP_status1_raddr               memory  0   00000064 0000ffff   1   0
```

```
ROP_dummy_cmd_addr              memory  0   00000066 0000ffff   1   1
ROP_dummy_raddr                 memory  0   00000068 0000ffff   1   0
ROP_LATDelayReg_addr            memory  0   00000070 0000ffff   1   1
ROP_VMEWriteAllMask_addr        memory  0   00000072 0000ffff   1   1
ROP_TestMask0                   memory  0   00000074 0000ffff   1   1
ROP_TestMask1                   memory  0   00000076 0000ffff   1   1
ROP_TestMask2                   memory  0   00000078 0000ffff   1   1
ROP_TestMask3                   memory  0   0000007A 0000ffff   1   1
ROP_TestMask4                   memory  0   0000007C 0000ffff   1   1
ROP_TestMask5                   memory  0   0000007E 0000ffff   1   1
ROP_TestMask6                   memory  0   00000080 0000ffff   1   1
ROP_TestMask7                   memory  0   00000082 0000ffff   1   1
ROP_DistrRam_base               memory  0   00000400 0000ffff   1   1
ROP_BlockRam_base               memory  0   00010000 0000ffff   1   1
*
* addresses extracted from InputFPGA/src/IFVMEAddrMap.vhd
* blue= new addresses Apr2009
*
-------------
IF_chip_id0_raddr               memory  0   00000000 0000ffff   1   0
IF_chip_id1_raddr               memory  0   00000002 0000ffff   1   0
IF_chip_rev0_raddr              memory  0   00000004 0000ffff   1   0
IF_chip_rev1_raddr              memory  0   00000006 0000ffff   1   0
*
* addresses 0x8 - 0xE used for date and time !!!
*
IF_SyncConfigReg_addr0          memory  0   00000020 0000ffff   1   1
IF_SyncConfigReg_addr1          memory  0   00000022 0000ffff   1   1
IF_SyncConfigReg_addr2          memory  0   00000024 0000ffff   1   1
IF_SyncConfigReg_addr3          memory  0   00000026 0000ffff   1   1
IF_ReadoutSyncReg_addr          memory  0   00000028 0000ffff   1   1
IF_LATDelayReg_addr             memory  0   00000030 0000ffff   1   1
IF_SimuSpyConfig_addr           memory  0   00000032 0000ffff   1   1
IF_SpyDepth_addr                memory  0   00000034 0000ffff   1   1
IF_SpyArmPulse_waddr            memory  0   00000036 0000ffff   0   1
IF_SpyDone_raddr                memory  0   00000038 0000ffff   1   0
IF_dummy_raddr                  memory  0   00000040 0000ffff   1   0
IF_PhaseMonitorStatusReg_raddr  memory  0   00000042 0000ffff   1   0
IF_ErrorMonitorStatusReg_raddr  memory  0   00000044 0000ffff   1   0
IF_CompareCounterReset_waddr    memory  0   00000046 0000ffff   0   1
IF_PhaseMonitorCounter_raddr0   memory  0   00000050 0000ffff   1   0
IF_PhaseMonitorCounter_raddr1   memory  0   00000060 0000ffff   1   0
IF_PhaseMonitorCounter_raddr2   memory  0   00000070 0000ffff   1   0
IF_PhaseMonitorCounter_raddr3   memory  0   00000080 0000ffff   1   0
IF_ErrorMonitorCounter_raddr0   memory  0   00000090 0000ffff   1   0
IF_ErrorMonitorCounter_raddr1   memory  0   000000A0 0000ffff   1   0
IF_ErrorMonitorCounter_raddr2   memory  0   000000B0 0000ffff   1   0
IF_ErrorMonitorCounter_raddr3   memory  0   000000C0 0000ffff   1   0
*
* blue= new addresses Jul2009
*
IF_rate_counterl_mu0            memory  0   000000D0 0000ffff   1   0
IF_rate_counterh_mu0            memory  0   000000D2 0000ffff   1   0
IF_rate_counterl_mu1            memory  0   000000D4 0000ffff   1   0
```

```
IF_rate_counterh_mu1           memory  0   000000D6 0000ffff   1   0
IF_rate_counterl_mu2           memory  0   000000D8 0000ffff   1   0
IF_rate_counterh_mu2           memory  0   000000DA 0000ffff   1   0
IF_rate_counterl_mu3           memory  0   000000DC 0000ffff   1   0
IF_rate_counterh_mu3           memory  0   000000DE 0000ffff   1   0

IF_lum_segm_period_l           memory  0   000000E0 0000ffff   1   1
IF_lum_segm_period_h           memory  0   000000E2 0000ffff   1   1
IF_lum_segm_number_raddr       memory  0   000000E4 0000ffff   1   0

IF_CompareCounterVec_raddr     memory  0   00000100 0000ffff   1   0
IF_DistrRam_base               memory  0   00000400 0000ffff   1   1
IF_BlockRam_base               memory  0   00010000 0000ffff   1   1
*
* addresses extracted from LogicFPGA/src/LFVMEAddrMap.vhd
*
LF_chip_id0_raddr              memory  0   00000000 0000ffff   1   0
LF_chip_id1_raddr              memory  0   00000002 0000ffff   1   0
LF_chip_rev0_raddr             memory  0   00000004 0000ffff   1   0
LF_chip_rev1_raddr             memory  0   00000006 0000ffff   1   0
LF_CDLConfig_addr0             memory  0   00000020 0000ffff   1   1
LF_CDLConfig_addr1             memory  0   00000022 0000ffff   1   1
LF_SortRankOffset_addr         memory  0   00000024 0000ffff   1   1
LF_MMConfig_SRK_addr           memory  0   00000026 0000ffff   1   1
LF_MMConfig_Phi_addr           memory  0   00000028 0000ffff   1   1
LF_MMConfig_Eta_addr           memory  0   0000002A 0000ffff   1   1
LF_MMConfig_Pt_addr            memory  0   0000002C 0000ffff   1   1
LF_MMConfig_Charge_addr        memory  0   0000002E 0000ffff   1   1
LF_MMConfig_MIP_addr           memory  0   00000030 0000ffff   1   1
LF_MMConfig_ISO_addr           memory  0   00000032 0000ffff   1   1
LF_dummy_raddr                 memory  0   00000034 0000ffff   1   0
LF_DistrRam_base               memory  0   00000400 0000ffff   1   1
LF_MatchQualLUT_base0          memory  0   00000400 0000ffff   1   1
LF_MatchQualLUT_base1          memory  0   00001400 0000ffff   1   1
LF_MatchQualLUT_base2          memory  0   00002400 0000ffff   1   1
LF_COUDeltaEtaLUT_base0        memory  0   00003400 0000ffff   1   1
LF_COUDeltaEtaLUT_base1        memory  0   00005400 0000ffff   1   1
LF_OvlEtaConvLUT_base0         memory  0   00007400 0000ffff   1   1
LF_OvlEtaConvLUT_base1         memory  0   00007600 0000ffff   1   1
LF_OvlEtaConvLUT_base2         memory  0   00007800 0000ffff   1   1
LF_EtaConvLUT_base0            memory  0   00007A00 0000ffff   1   1
LF_EtaConvLUT_base1            memory  0   00007C00 0000ffff   1   1
LF_MergeRankPtQLUT_base0       memory  0   00007E00 0000ffff   1   1
LF_MergeRankPtQLUT_base1       memory  0   00008600 0000ffff   1   1
LF_PhiProEtaConvLUT_base0      memory  0   00008E00 0000ffff   1   1
LF_PhiProEtaConvLUT_base1      memory  0   00009000 0000ffff   1   1
LF_BlockRam_base               memory  0   00010000 0000ffff   1   1
LF_SortRankEtaQLUT_base0       memory  0   00010000 0000ffff   1   1
LF_SortRankEtaQLUT_base1       memory  0   00010400 0000ffff   1   1
LF_SortRankPtQLUT_base0        memory  0   00010800 0000ffff   1   1
LF_SortRankPtQLUT_base1        memory  0   00010C00 0000ffff   1   1
LF_SortRankEtaPhiLUT_base0     memory  0   00011000 0000ffff   1   1
LF_SortRankEtaPhiLUT_base1     memory  0   00015000 0000ffff   1   1
LF_SortRankCombineLUT_base0    memory  0   00019000 0000ffff   1   1
LF_SortRankCombineLUT_base1    memory  0   0001B000 0000ffff   1   1
```

```
LF_DeltaEtaLUT_base0            memory  0   0001D000 0000ffff   1   1
LF_PtMixLUT_base0               memory  0   00025000 0000ffff   1   1
LF_MergeRankEtaQLUT_base0       memory  0   00026000 0000ffff   1   1
LF_MergeRankEtaQLUT_base1       memory  0   00026800 0000ffff   1   1
LF_MergeRankEtaPhiLUT_base0     memory  0   00027000 0000ffff   1   1
LF_MergeRankEtaPhiLUT_base1     memory  0   00029000 0000ffff   1   1
LF_MergeRankCombineLUT_base0    memory  0   0002B000 0000ffff   1   1
LF_MergeRankCombineLUT_base1    memory  0   0002C000 0000ffff   1   1
LF_DisableHotLUT_base0          memory  0   0002D000 0000ffff   1   1
LF_PhiProLUT_base0              memory  0   0002F000 0000ffff   1   1
LF_PhiProLUT_base1              memory  0   00031000 0000ffff   1   1
*
* addresses extracted from MipIsoAU/src/MIAUVMEAddrMap.vhd
*
MIAU_chip_id0_raddr             memory  0   00000000 0000ffff   1   0
MIAU_chip_id1_raddr             memory  0   00000002 0000ffff   1   0
MIAU_chip_rev0_raddr            memory  0   00000004 0000ffff   1   0
MIAU_chip_rev1_raddr            memory  0   00000006 0000ffff   1   0
MIAU_dummy_raddr                memory  0   00000020 0000ffff   1   0
*
* not implemented in firmware yet
*MIAU_bcreset_long_delay        memory  0   00000024 0000ffff   1   1
*
MIAU_ReadoutSyncReg_addr        memory  0   00000028 0000ffff   1   1
MIAU_SimuSpyConfig_addr         memory  0   00000032 0000ffff   1   1
MIAU_SpyDepth_addr              memory  0   00000034 0000ffff   1   1
MIAU_SpyArmPulse_waddr          memory  0   00000036 0000ffff   0   1
MIAU_SpyDone_raddr              memory  0   00000038 0000ffff   1   0
*
* not implemented in firmware yet
*MIAU_SelMQ_PSBchan_addr        memory  0   0000003A 0000ffff   1   1
*
MIAU_DistrRam_base              memory  0   00000400 0000ffff   1   1
MIAU_EtaConvLUT_base0           memory  0   00000400 0000ffff   1   1
MIAU_EtaConvLUT_base1           memory  0   00000600 0000ffff   1   1
MIAU_EtaConvLUT_base2           memory  0   00000800 0000ffff   1   1
MIAU_EtaConvLUT_base3           memory  0   00000A00 0000ffff   1   1
MIAU_SimuRAM_base               memory  0   00002000 0000ffff   1   1
*
* not implemented in firmware yet
*MIAU_SpyRAM_base               memory  0   00005000 0000ffff   1   1
*
MIAU_BlockRam_base              memory  0   00010000 0000ffff   1   1
MIAU_PhiPro1LUT_base0           memory  0   00010000 0000ffff   1   1
MIAU_PhiPro1LUT_base1           memory  0   00014000 0000ffff   1   1
MIAU_PhiPro1LUT_base2           memory  0   00018000 0000ffff   1   1
MIAU_PhiPro1LUT_base3           memory  0   0001C000 0000ffff   1   1
MIAU_PhiPro2LUT_base0           memory  0   00020000 0000ffff   1   1
MIAU_PhiPro2LUT_base1           memory  0   00024000 0000ffff   1   1
MIAU_PhiPro2LUT_base2           memory  0   00028000 0000ffff   1   1
MIAU_PhiPro2LUT_base3           memory  0   0002C000 0000ffff   1   1
MIAU_EtaProLUT_base0            memory  0   00030000 0000ffff   1   1
MIAU_EtaProLUT_base1            memory  0   00038000 0000ffff   1   1
MIAU_EtaProLUT_base2            memory  0   00040000 0000ffff   1   1
```

```
MIAU_EtaProLUT_base3            memory  0   00048000 0000ffff   1   1
*
* addresses extracted from SortFPGA/src/SFVMEAddrMap.vhd
*
SF_chip_id0_raddr               memory  0   00000000 0000ffff   1   0
SF_chip_id1_raddr               memory  0   00000002 0000ffff   1   0
SF_chip_rev0_raddr              memory  0   00000004 0000ffff   1   0
SF_chip_rev1_raddr              memory  0   00000006 0000ffff   1   0
SF_ReadoutSyncReg_addr          memory  0   00000020 0000ffff   1   1
SF_LATDelayReg_addr             memory  0   00000022 0000ffff   1   1
SF_SimuSpyConfig_addr           memory  0   00000024 0000ffff   1   1
SF_SpyDepth_addr                memory  0   00000026 0000ffff   1   1
SF_SpyArmPulse_waddr            memory  0   00000028 0000ffff   0   1
SF_SpyDone_raddr                memory  0   00000030 0000ffff   1   0
SF_dummy_raddr                  memory  0   00000032 0000ffff   1   0
SF_DistrRam_base                memory  0   00000400 0000ffff   1   1
SF_BlockRam_base                memory  0   00010000 0000ffff   1   1
```

4.2 VME_ROP chip

Chip-Identifier, Revision number, Date, Time
Registers to identify the firmware.

```
ROP_chip_id0_raddr              memory  0   00000000 0000ffff   1   0
ROP_chip_id1_raddr              memory  0   00000002 0000ffff   1   0
ROP_chip_rev0_raddr             memory  0   00000004 0000ffff   1   0
ROP_chip_rev1_raddr             memory  0   00000006 0000ffff   1   0
```

```
ROP_chip_id0_raddr      00000000    // identifier low word
ROP_chip_id1_raddr      00000002    // identifier high word
ROP_chip_rev0_raddr     00000004    // revision-nr low word
ROP_chip_rev1_raddr     00000006    // revision-nr high word
ROP_chip_month_day      00000008    // month & day          --not defined in Address table
ROP_chip_year_raddr     0000000A    // year 4 char          -- not defined in Address table
ROP_chip_hour_min       0000000C    // hour & min           -- not defined in Address table
ROP_chip_ffff_sec       0000000E    // ffff & sec           -- not defined in Address table
```

JTAG registers: 20….2E

```
ROP_JTAG_base                   memory  0   00000020 0000ffff   1   1
```

--Others registers are accessed with appropriate offset

Reset commands

```
ROP_reset_addr                  memory  0   00000030 0000ffff   1   1
```

Set the reset_xxx =1 then set reset_xxx =0 to make pulse. Therefore the length is defined by software.

Bit 15 -10: not used
Bit 9: 1= reset_SRT
Bit 8: 1= reset_AUB
Bit 7: 1= reset_LFB
Bit 6: 1= reset_LFF
Bit 5: 1= reset_AUF

Bit 4: 1= reset_INB
Bit 3: 1= reset_IND
Bit 2: 1= reset_INC
Bit 1: 1= reset_INF
Bit 0: 1= reset_ROP ← is not used (would reset the logic inside the ROP chip).
See (ROP_chip.vhd)

```
ROP_reset_dcm_addr              memory  0   00000032 0000ffff   1   1
```

Set the reset_dcm_xxx =1 then set reset_dcm_xxx =0 to make pulse. Therefore the length is defined by software.
Bit 15 -10: not used
Bit 9: 1= reset_dcm_SRT
Bit 8: 1= reset_dcm_AUB
Bit 7: 1= reset_dcm_LFB
Bit 6: 1= reset_dcm_LFF
Bit 5: 1= reset_dcm_AUF
Bit 4: 1= reset_dcm_INB
Bit 3: 1= reset_dcm_IND
Bit 2: 1= reset_dcm_INC
Bit 1: 1= reset_dcm_INF
Bit 0: 1= reset_dcm_ROP ← is not used (would reset the logic inside the ROP chip)
bit0: reset_dcm_rop<='0'...to avoid Rop crash.
See (ROP_chip.vhd)

Command register

```
ROP_command_addr                memory  0   00000034 0000ffff   1   1
```

set GMT-status to busy, ready for TCS board

```
bit 1: =1 GMT_is_busy
bit 0: =1 GMT_is_ready
```

See ROPBoardControl.vhd

```
ROP_JTAG_enable_addr            memory  0   00000036 0000ffff   1   1
```

-- bit 0 : =1 enable Xilinx jtag-chain,
-- bit 1 : =1 enable Altera jtag-chain,

Configuration addresses

-- *Configuration of FPGA*
-- ***nprog*** *is used to start programming*
-- *rising edge on prog_b signal of the FPGA starts configuration*
-- *the FPGA then drives the signal init_b low while it clears the configuration.*
-- *Configuration can be delayed by driving init_b low externally.*
-- *Configuration mode pins are sampled when when init_b transitions to high*
-- *afterwards, configuration via clock and data starts*

```
ROP_prog_enable_addr            memory  0   00000040 0000ffff   1
1
```

enable programming of chips
Loading a '1' enables reloading the addressed FPGA from the PROMs.
Bit 15 -10: not defined
Bit 9: en_progr_SRT

Bit 8: en_progr_AUB
Bit 7: en_progr_LFB
Bit 6: en_progr_LFF
Bit 5: en_progr_AUF
Bit 4: en_progr_INB
Bit 3: en_progr_IND
Bit 2: en_progr_INC
Bit 1: en_progr_INF
Bit 0: en_progr_ROP ← is not used (no NPROG on ROP chip)

`ROP_nprog_addr  memory  0  00000042 0000ffff  1  1`

NPROG signal to load configuration data from Proms into the FPGA chips
Set the nprog_xxx =0, then nprog_xxx =1 to makea low active pulse program pulse to reload the FPGA from the PROMs. Therefore the length is defined by software.
➔ *Software should send 2 consecutive write commands, the first with the bit=0 for the addressed chip, the second with all bits =1.*

Bit 15 -10: 1111_11
Bit 9: nprog_SRT
Bit 8: nprog_AUB
Bit 7: nprog_LFB
Bit 6: nprog_LFF
Bit 5: nprog_AUF
Bit 4: nprog_INB
Bit 3: nprog_IND
Bit 2: nprog_INC
Bit 1: nprog_INF
Bit 0: nprog_ROP ← is not used (no NPROG on ROP chip, no pin for this signal)

`ROP_init_cmd_addr  memory  0  00000044 0000ffff  1  1`

NINIT signal to FPGA chips:
Make pulse by software: 'Z'--'0'--'Z' :
a.) write a '1' to the register ➔ forces ninit signal to low then
b.) write a '0' to the register ➔ releases ninit signal to high Z
Bit 15 -10: not assigned
Bit 9: do_init_SRT
Bit 8: do_init_AUB
Bit 7: do_init_LFB
Bit 6: do_init_LFF
Bit 5: do_init_AUF
Bit 4: do_init_INB
Bit 3: do_init_IND
Bit 2: do_init_INC
Bit 1: do_init_INF
Bit 0: do_init_ROP ← is not used (no NPROG on ROP chip, no pin for this signal)

➔ *Software should send 2 consecutive write commands, the first with the bit set for the*

addressed chip, the second with all bits =0.

```
ROP_init_status_raddr     memory  0   00000046 0000ffff   1   0
```

INIT status of FPGA chips: The inverted value of the configuration signals 'NINIT' is shown when reading this register.
Bit 15 -10: 0000_00
Bit 9: init_SRT
Bit 8: init_AUB
Bit 7: init_LFB
Bit 6: init_LFF
Bit 5: init_AUF
Bit 4: init_INB
Bit 3: init_IND
Bit 2: init_INC
Bit 1: init_INF
Bit 0: init_ROP ← is not used (no NPROG on ROP chip, no pin for this signal)

```
ROP_done_raddr            memory  0   00000048 0000ffff   1   0
```

DONE flags of FPGA chips, =1 when configuration has been done
Bit 15 -10: 0000_00
Bit 9: done_SRT
Bit 8: done_AUB
Bit 7: done_LFB
Bit 6: done_LFF
Bit 5: done_AUF
Bit 4: done_INB
Bit 3: done_IND
Bit 2: done_INC
Bit 1: done_INF
Bit 0: done_ROP ← is not used (no NPROG on ROP chip, no pin for this signal)

DIN-addresses for serial configuration data

```
ROP_din_INF_addr          memory  0   0000004A 0000ffff   1   1
ROP_din_INC_addr          memory  0   0000004C 0000ffff   1   1
ROP_din_IND_addr          memory  0   0000004E 0000ffff   1   1
ROP_din_INB_addr          memory  0   00000050 0000ffff   1   1
ROP_din_AUF_addr          memory  0   00000052 0000ffff   1   1
ROP_din_LFF_addr          memory  0   00000054 0000ffff   1   1
ROP_din_LFB_addr          memory  0   00000056 0000ffff   1   1
ROP_din_AUB_addr          memory  0   00000058 0000ffff   1   1
ROP_din_SRT_addr          memory  0   0000005A 0000ffff   1   1
```

DIN registers generate a clock pulse when written, only bit 0 is used. The pulse is sent to the DIN pin of the addressed FPGA (INF….).
ROP chip sends the data bit via DIN (tristate) outputs to other chips and concurrently a short CCLK pulse. The tristate outputs(DIN, CCLK pins) are switched on by corresponding bits of the ROP_prog_enable_addr register.

See ROPBoardControl.vhd, ROPProgPulseRegister.vhd

```
ROP_dcm_locked_raddr        memory  0   00000060 0000ffff   1   0
```

LOCKED flags from FPGA-chips show if all FPGAs are running with clock.
Bit 15 -10: 0000_00
Bit 9: dcm_locked_SRT(0)
Bit 8: dcm_locked_AUB(0)
Bit 7: dcm_locked_LFB(0)
Bit 6: dcm_locked_LFF(0)
Bit 5: dcm_locked_AUF(0)
Bit 4: dcm_locked_INB(0)
Bit 3: dcm_locked_IND(0)
Bit 2: dcm_locked_INC(0)
Bit 1: dcm_locked_INF(0)
Bit 0: dcm_locked_ROP(0)

STATUS registers

	Status1	Status0
disconnected	0	0
ready	0	1
busy	1	0
Error ??	1	1

```
ROP_status0_raddr         memory  0   00000062 0000ffff   1   0
```

Bit 15 -10: 0000_00
Bit 9: status0_SRT(0)
Bit 8: status0_AUB(0)
Bit 7: status0_LFB(0)
Bit 6: status0_LFF(0)
Bit 5: status0_AUF(0)
Bit 4: status0_INB(0)
Bit 3: status0_IND(0)
Bit 2: status0_INC(0)
Bit 1: status0_INF(0)
Bit 0: status0_ROP(0)

See ROPBoardControl.vhd

```
ROP_status1_raddr         memory  0   00000064 0000ffff   1   0
```

Bit 15 -10: 0000_00
Bit 9: status1_SRT(1)
Bit 8: status1_AUB(1)
Bit 7: status1_LFB(1)
Bit 6: status1_LFF(1)
Bit 5: status1_AUF(1)
Bit 4: status1_INB(1)

Bit 3: status1_IND(1)
Bit 2: status1_INC(1)
Bit 1: status1_INF(1)
Bit 0: status1_ROP(1)

See ROPBoardControl.vhd

Command register

```
ROP_dummy_cmd_addr              memory  0   00000066 0000ffff   1   1
```

Bit 15 - 12: not used
Bit 11: 1= send BCRES from TIM to AUB chip, 0= send bit8 to AUB
Bit 10: 1= send BCRES from TIM to AUF chip, 0= send bit5 to AUF
Bit 9: dummy_SRT
Bit 8: VME_BCRES_to_AUB *//(was dummy_AUB)*
Bit 7: dummy_LFB
Bit 6: dummy_LFF
Bit 5: VME_BCRES_to_AUF *//(was dummy_AUF)*
Bit 4: dummy_INB
Bit 3: dummy_IND
Bit 2: dummy_INC
Bit 1: dummy_INF
Bit 0: dummy_ROP
To make BCRES pulse by software first send '1' then '0' to AUF, AUB chip. (X"0120", then X"0000").
To run with BCRES from TIM load X"0C00".

```
ROP_dummy_raddr                 memory  0   00000068 0000ffff   1   0
```

Latency delay

```
ROP_LATDelayReg_addr            memory  0   00000070 0000ffff   1   1
```

See BXReadCounter module that is used in ROPROP.

Bit 15: 0= 3bx per event, 1= 5 bx per event
Bits 7-4: LAT_delay2
Bits 3-0: LAT_delay1

The total latency = LAT_delay2 + LAT_delay1 (plus inherent 2bx delay).
The latency is used to reset the readout counter later then the write counter to consider the local latency of the returning L1A signal. Readout of data starts at the bcnr = bcnr(L1A) – 1 to get data from bunch crossings (L1A-1, L1A, L1A+1).
When running in 5bx per event mode then the firmware delays the resetting bcres signal by one additional bx to start the readout of data at bc-nr = bcnr(L1A) – 2 to get data from bunch crossings (L1A-2, L1A-1, L1A, L1A+1, L1A+2).

HB 2014-04-30:
ROP => v1006
Added one SRL16 for latency delay with fixed value of 16, to get greater delays in module BXReadCounter.vhd, because L1A from TCDS has more latency (than from TCS).

v1006 => **Total delay = Latency_delay2 + Latency_delay1 + 2 + 16**

Mask

```
ROP_VMEWriteAllMask_addr        memory  0   00000072 0000ffff   1   1
```

Mask defines to which FPGAs a common VME command is directed.
Power-up value: = "0000000000011110" =hex001E, -- default: all four input chips …to write to all chips with the same instruction.

-- Send VME command concurrently to chips: use it only for IN chips
-- because they have got the same registers
-- Default: to all 4 IN chips(same firmware): "001e"

TestMasks

Testmasks bits define which signals are applied to the corresponding testpoint.
If more bits are set to 1 then the 'OR' of the signals will be shown.
The testpoints are accessible on the bottom side of the GMT board behind the ROP chip.

```
ROP_TestMask0                   memory  0   00000074 0000ffff   1   1
ROP_TestMask1                   memory  0   00000076 0000ffff   1   1
ROP_TestMask2                   memory  0   00000078 0000ffff   1   1
ROP_TestMask3                   memory  0   0000007A 0000ffff   1   1
ROP_TestMask4                   memory  0   0000007C 0000ffff   1   1
ROP_TestMask5                   memory  0   0000007E 0000ffff   1   1
ROP_TestMask6                   memory  0   00000080 0000ffff   1   1
ROP_TestMask7                   memory  0   00000082 0000ffff   1   1
```

ROP_TestMask0

Bit 15: ro_fetch_srt // fetch data from SRT chip
Bit 14: ro_fetch_inf // fetch data from INF chip
Bit 13: ro_fetch_ind
Bit 12: ro_fetch_inc
Bit 11: ro_fetch_inb
Bit10: vme_wr_ROP // VME_WRITE from VME interface to internal ROP logic
Bit 9: vme_en_auf // VME_EN (50ns) to FPGA chip
Bit 8: vme_en_aub
Bit 7: vme_en_lff
Bit 6: vme_en_lfb
Bit 5: vme_en_inf
Bit 4: vme_en_ind
Bit 3: vme_en_inc
Bit 2: vme_en_inb
Bit 1: vme_en_srt
Bit 0: vme_en_rop // VME_EN from VME interface to internal ROP logic

ROP_TestMask1 Bit 15: ch_link1(27) //channel link header bits
Bit 14: ch_link1(26)
Bit 13: ch_link1(25)
Bit 12: ch_link1(24)
Bit 11: rd_fifo // common read FIFO signal to all IN, SRT chips
Bit 10: DTACK_EXT // DTACK signal from ROP to VME64
Bit 9: ndtack_auf // NDTACK signal from FPGA chip to ROP
Bit 8: ndtack_aub
Bit 7: ndtack_lff

Bit 6: ndtack_lfb
Bit 5: ndtack_inf
Bit 4: ndtack_ind
Bit 3: ndtack_inc
Bit 2: ndtack_inb
Bit 1: ndtack_srt
Bit 0: ndtack_rop

ROP_TestMask2: bits 15-4 not used
Bit(15 …0): ro_data_INB(15 …0)
ROP_TestMask3: bits 15-4 not used
Bit(15 …0): ro_data_INC(15 …0)
ROP_TestMask4:
Bit(15 …0): ro_data_IND(15 …0)
ROP_TestMask5:
Bit(15 …0): ro_data_INF(15 …0)
ROP_TestMask6:
Bit(15 …0): ro_data_SRT(15 …0)
ROP_TestMask7:
Bits 15-8: ro_data_INB(23 …16)
Bits 7- 0: ro_data_SRT(23 …16)

RAM addresses

```
ROP_DistrRam_base              memory  0   00000400 0000ffff   1   1
ROP_BlockRam_base              memory  0   00010000 0000ffff   1   1
```

4.2.1 Remarks to ROP firmware

VME64-VMEin ROP: DSCYC from VME64 is used without an input-FF (original Design of H.S). In case of problems I'll add an IFD in ROPChip.vhd.
RO-bus: GMT does not use the ro-bus signals (2.May)

4.3 INPUT chips

```
IF_chip_id0_raddr              memory  0   00000000 0000ffff   1   0
IF_chip_id1_raddr              memory  0   00000002 0000ffff   1   0
IF_chip_rev0_raddr             memory  0   00000004 0000ffff   1   0
IF_chip_rev1_raddr             memory  0   00000006 0000ffff   1   0

IF_rate_counter_raddr           00000008  … base address for 8 registers
IF_rate_counterl_mu0           memory  0   000000D0 0000ffff   1   0
IF_rate_counterh_mu0           memory  0   000000D2 0000ffff   1   0
IF_rate_counterl_mu1           memory  0   000000D4 0000ffff   1   0
IF_rate_counterh_mu1           memory  0   000000D6 0000ffff   1   0
IF_rate_counterl_mu2           memory  0   000000D8 0000ffff   1   0
IF_rate_counterh_mu2           memory  0   000000DA 0000ffff   1   0
IF_rate_counterl_mu3           memory  0   000000DC 0000ffff   1   0
IF_rate_counterh_mu3           memory  0   000000DE 0000ffff   1   0

IF_lum_segm_period_l           memory  0   000000E0 0000ffff   1   1
IF_lum_segm_period_h           memory  0   000000E2 0000ffff   1   1
```

```
IF_lum_segm_number_raddr        memory  0   000000E4 0000ffff   1   0
```

IF_SyncConfigReg_addr 00000020 … *base address for 4 registers*

```
IF_SyncConfigReg_addr0          memory  0   00000020 0000ffff   1   1
     For Muon0:
      15 – 13: not used
      12: clr_on_parity_error  // 1➔ error clears muons
      11: clr_on_bxerr
      10: clr_on_syerr
      9: use_SLR        // 0 ➔ min delay=0, 1 ➔ min delay =1 from SLR
      8: use_delay
      7  -4: delay in SLR  // 0…15
      3: sel_sync_phase3  //late
      2: sel_sync_phase2  //
      1: sel_sync_phase1  //
      0: sel_sync_phase0  //early

IF_SyncConfigReg_addr1          memory  0   00000022 0000ffff   1   1
      same for Muon1
IF_SyncConfigReg_addr2          memory  0   00000024 0000ffff   1   1
      same for Muon2
IF_SyncConfigReg_addr3          memory  0   00000026 0000ffff   1   1
      same for Muon3
```

IF_ReadoutSyncReg_addr memory 0 00000028 0000ffff 1 1

IF_LATDelayReg_addr memory 0 **00000030** 0000ffff 1 1

Used in module 'BXReadCounter' to reset the BC-Counter for the Readout logic.
In 3BC_mode reading starts one address before the triggering BC.
In 5BC_mode the BC counter is cleared 1 BC later to start reading 2 addresses before the triggering BC.
Bit 15: 1= 5_BC readout mode; 0= 3_BC readout mode
Bits 7 – 4 : Latency_delay2
Bits 3 – 0 : Latency_delay1

Total delay = Latency_delay2 + Latency_delay1 + 2

Unit= 1BC(25 ns)

HB 2014-04-30:
INx => v1008
Added one SRL16 for latency delay with fixed value of 16, to get greater delays in module BXReadCounter_IN_SRT.vhd, because L1A from TCDS has more latency (than from TCS).

v1008 => **Total delay = Latency_delay2 + Latency_delay1 + 2 + 16**

IF_SimuSpyConfig_addr memory 0 **00000032** 0000ffff 1 1

bit15-2: not used
bit1: DummyIs BCReset // 1=use_dummy_as_bcreset; 0=use BCReset
bit0: SimuMode // 1= sim_mode, 0= spy mode

When sim_mode=1 then simulation data are being sent always from the memories to the Logic chips as long as periodic BCRes arrives in the IN chip. In async mode when using dummy_pulse probably only one cycle runs.

```
IF_SpyDepth_addr                memory  0   00000034 0000ffff   1   1
      bit11-0: define length of SPY memory

IF_SpyArmPulse_waddr            memory  0   00000036 0000ffff   0   1
```
bit0: Writing '1' makes a pulse that sets a 'spy_armed' FF so that that spying is started with the next BCReset signal (or with next dummy_pulse). The FF is cleared afterwards by the 2nd BCreset. Spying ends after nnn words as defined by SpyDepth register.

```
IF_SpyDone_raddr                memory  0   00000038 0000ffff   1   0
```
bit0: Spy_Done After the spying the Done FF is set.

```
IF_dummy_raddr                  memory  0   00000040 0000ffff   1   0
IF_PhaseMonitorStatusReg_raddr memory  0   00000042 0000ffff   1   0
IF_ErrorMonitorStatusReg_raddr memory  0   00000044 0000ffff   1   0
      bit3: clear_muon
      bit2: parity_err
      bit1: bx_err
      bit0: sync_err

IF_CompareCounterReset_waddr    memory  0   00000046 0000ffff   0   1

IF_PhaseMonitorCounter_raddr0   memory  0   00000050 0000ffff   1   0
IF_PhaseMonitorCounter_raddr1   memory  0   00000060 0000ffff   1   0
IF_PhaseMonitorCounter_raddr2   memory  0   00000070 0000ffff   1   0
IF_PhaseMonitorCounter_raddr3   memory  0   00000080 0000ffff   1   0
IF_ErrorMonitorCounter_raddr0   memory  0   00000090 0000ffff   1   0
IF_ErrorMonitorCounter_raddr1   memory  0   000000A0 0000ffff   1   0
IF_ErrorMonitorCounter_raddr2   memory  0   000000B0 0000ffff   1   0
IF_ErrorMonitorCounter_raddr3   memory  0   000000C0 0000ffff   1   0

IF_CompareCounterVec_raddr      memory  0   00000100 0000ffff   1   0
```
Undelayed input data are compared with simulated data. Any difference increments a counter.

```
IF_DistrRam_base                memory  0   00000400 0000ffff   1   1
IF_BlockRam_base                memory  0   00010000 0000ffff   1   1
```

Logic remarks:
Empty muons :if pt = 1f or phi = ff =forbidden code
The rising edge of the `dummy' signal makes a 'dummy_pulse' that is used instead of a BCRES signal if 'DummyIs BCReset' =1.
Readout data:
bit31-30: ID of chip INB= , INC= , IND= , INF=
bit 29-0: input muon

4.4 LOGIC chips

4.5 ASSIGNMENT UNIT chips

Version V0002 Jan 2009
- SimSpy memory replaces Simulation memory
- Selection logic to connects either MIP or QUIET bits from one of the PSB channels to the SimSpy memory. → new register added

MIAU_bcreset_long_delay addr = hex 24
Bit 11 – 0 : bcres_delay range 0 …3563 dec (= 0 … 0DEB hex); unit = 1BC
Default value = X"0DEA" -2BC to apply Simulation data correctly with ROP Version of 2009
Zero_value = X"FFFF" … delay = 0

MIAU_SelMQ_PSBchan_addr addr = hex 3A

Default: = X"0000" → select QUIET bits of PSB19_chan0

Bit8 = SEL_MIP_bits =1 selects MIP bits; =0 selects QUIET bits
Bit 7 – 0:
= X"00" selects PSB19_chan0
= X"01" selects PSB19_chan1
= X"02" selects PSB19_chan2
= X"03" selects PSB19_chan3
= X"04" selects PSB19_chan4
= X"05" selects PSB19_chan5
= X"06" selects PSB19_chan6
= X"07" selects PSB19_chan7
= X"10" selects PSB20_chan0
= X"11" selects PSB20_chan1
= X"12" selects PSB20_chan2
= X"13" selects PSB20_chan3
= X"14" selects PSB20_chan4
= X"15" selects PSB20_chan5
= X"16" selects PSB20_chan6
= X"17" selects PSB20_chan7
= X"20" selects PSB21_chan0
= X"21" selects PSB21_chan1
= X"22" selects PSB21_chan2
= X"23" selects PSB21_chan3
= X"24" selects PSB21_chan4
= X"25" selects PSB21_chan5
= X"26" selects PSB21_chan6
= X"27" selects PSB21_chan7

4.6 SORTER chip

```
SF_chip_id0_raddr              memory  0   00000000 0000ffff   1   0
SF_chip_id1_raddr              memory  0   00000002 0000ffff   1   0
SF_chip_rev0_raddr             memory  0   00000004 0000ffff   1   0
SF_chip_rev1_raddr             memory  0   00000006 0000ffff   1   0
SF_ReadoutSyncReg_addr         memory  0   00000020 0000ffff   1   1
SF_LATDelayReg_addr            memory  0   00000022 0000ffff   1   1
SF_SimuSpyConfig_addr          memory  0   00000024 0000ffff   1   1
SF_SpyDepth_addr               memory  0   00000026 0000ffff   1   1
SF_SpyArmPulse_waddr           memory  0   00000028 0000ffff   0   1
SF_SpyDone_raddr               memory  0   00000030 0000ffff   1   0
SF_dummy_raddr                 memory  0   00000032 0000ffff   1   0
SF_DistrRam_base               memory  0   00000400 0000ffff   1   1
SF_BlockRam_base               memory  0   00010000 0000ffff   1   1
```

SF_ReadoutSyncReg_addr
base + `00000020`
Used in module 'BXCounter'
Bit 4:
1= use bcres_delay; minimum delay = 1 BC
0= do not use bcres_delay ➔ delay = 0 … minimum
Bits 3 – 0 : bcres_delay
Example: …1_0000 ➔ total delay = 1 BC
…0_0000 ➔ total delay = 0 BC

BCRESET CHAIN:
BCRES_SRT or dummy_pulse → ReadOutLogic
→ | bcres_delay| → → → → → addr_cntr of Simulation RAM.
|→ FF → FF → → main BC counter
|→ → SimuSpyLogic

dummy_pulse…from ROP chip to SRT and all IN chips.

SF_LATDelayReg_addr
base + `00000022`
Used in module 'BXReadCounter' to reset the BC-Counter for the Readout logic.
In 3BC_mode reading starts one address before the triggering BC.
In 5BC_mode the BC counter is cleared 1 BC later to start reading 2 addresses before the triggering BC.
Bit 15: 1= 5_BC readout mode; 0= 3_BC readout mode
Bits 7 – 4 : Latency_delay2
Bits 3 – 0 : Latency_delay1
Total delay = Latency_delay2 + Latency_delay1 + 2
Unit= 1BC(25 ns)

HB 2014-04-30:
SORT => v1003
Added one SRL16 for latency delay with fixed value of 16, to get greater delays in module

BXReadCounter_IN_SRT.vhd, because L1A from TCDS has more latency (than from TCS).
v1003 => **Total delay = Latency_delay2 + Latency_delay1 + 2 + 16**

SF_SimuSpyConfig_addr
base + `00000024`
bit1: 1= use_dummy_as_bcreset 0 = use BCRES signal from backplane
bit0: 1 = simulation mode 0 = spy_mode

SF_ SpyDepth _addr
base + 00000026
The spy memory will be filled up to the 'SpyDepth' limit, which defines the last spy address in 32-bit words. Should be less than 1 k.

SF_SpyArmPulse_waddr
base + 00000028
bit0: 1 = 'arming spy logic' pulse
Spy_logic is armed with a pulse to start spying with next BCRESET.
Spying stops at the 2nd BCRESET or at the Spy_depth limit.
VHDL comment: *--* Spy is armed with a pulse*
-- write_enable is set at next bcreset*
-- write enable is cleared at 2nd bcreset or counter full*

SF_SpyDone_raddr
base + 00000030
When spying has been done the Spy_logic sets bit0 := 1.
The 'arming spy logic' pulse clears the SpyDone bit0.

SF_dummy_raddr
base + 00000030
Don't care about the content of this register. It is used to keep unused io-pads.

SIM/SPY memory
`SF_BlockRam_base 00010000`
0 ..3 muons to GT; 4 = SRTRanks
All simulation/spy RAMs are 32 bit wide, 4k longwords deep, i.e. they occupy 16kbyte address space (4000 hex).
There are 4 Simu/Spy RAMS in each Input FPGA and 5 in the Sort FPGA.
*Attention: With a VirtexII 2000 the Simu/SpyRAM in the Sort FPGA is only 1k*32 bits deep, but the address space is defined as for 4k memories with 3k gaps between.*
Address ranges:
Muon0 memory: `00010000` … `00010FFE`
Muon1 memory: `00014000` … `00014FFE`
Muon2 memory: `00018000` … `00014FFE`
Muon3 memory: `0001C000` … `0001CFFE`
RANK memory: `00020000` … `00020FFE`

5 Test Procedures

6 MIP/ISO bits from GCT➔PSB Cables

6.1 Mapping GCT cables

		-6	-5	-4	-3	-2	-1	0
phi	0	3_6_0	3_45_1	3_45_0	1_23_1	1_23_0	1_01_1	1_01_0
	1	3_6_1	3_45_3	3_45_2	1_23_3	1_23_2	1_01_3	1_01_2
	2	3_6_8	3_45_9	3_45_8	1_23_9	1_23_8	1_01_9	1_01_8
	3	3_6_9	3_45_11	3_45_10	1_23_11	1_23_10	1_01_11	1_01_10
	4	4_6_4	4_45_5	4_45_4	2_23_5	2_23_4	2_01_5	2_01_4
	5	4_6_5	4_45_7	4_45_6	2_23_7	2_23_6	2_01_7	2_01_6
	6	7_6_0	7_45_1	7_45_0	5_23_1	5_23_0	5_01_1	5_01_0
	7	7_6_1	7_45_3	7_45_2	5_23_3	5_23_2	5_01_3	5_01_2
	8	7_6_8	7_45_9	7_45_8	5_23_9	5_23_8	5_01_9	5_01_8
	9	7_6_9	7_45_11	7_45_10	5_23_11	5_23_10	5_01_11	5_01_10
	10	8_6_4	8_45_5	8_45_4	6_23_5	6_23_4	6_01_5	6_01_4
	11	8_6_5	8_45_7	8_45_6	6_23_7	6_23_6	6_01_7	6_01_6
	12	11_6_0	11_45_1	11_45_0	9_23_1	9_23_0	9_01_1	9_01_0
	13	11_6_1	11_45_3	11_45_2	9_23_3	9_23_2	9_01_3	9_01_2
	14	11_6_8	11_45_9	11_45_8	9_23_9	9_23_8	9_01_9	9_01_8
	15	11_6_9	11_45_11	11_45_10	9_23_11	9_23_10	9_01_11	9_01_10
	16	12_6_4	12_45_5	12_45_4	10_23_5	10_23_4	10_01_5	10_01_4
	17	12_6_5	12_45_7	12_45_6	10_23_7	10_23_6	10_01_7	10_01_6

0	1	2	3	4	5	6
1_01_4	1_01_5	1_23_4	1_23_5	3_45_4	3_45_5	3_6_4
1_01_6	1_01_7	1_23_6	1_23_7	3_45_6	3_45_7	3_6_5
2_01_0	2_01_1	2_23_0	2_23_1	4_45_0	4_45_1	4_6_0
2_01_2	2_01_3	2_23_2	2_23_3	4_45_2	4_45_3	4_6_1
2_01_8	2_01_9	2_23_8	2_23_9	4_45_8	4_45_9	4_6_8
2_01_10	2_01_11	2_23_10	2_23_11	4_45_10	4_45_11	4_6_9

5_01_4	5_01_5	5_23_4	5_23_5	7_45_4	7_45_5	7_6_4
5_01_6	5_01_7	5_23_6	5_23_7	7_45_6	7_45_7	7_6_5
6_01_0	6_01_1	6_23_0	6_23_1	8_45_0	8_45_1	8_6_0
6_01_2	6_01_3	6_23_2	6_23_3	8_45_2	8_45_3	8_6_1
6_01_8	6_01_9	6_23_8	6_23_9	8_45_8	8_45_9	8_6_8
6_01_10	6_01_11	6_23_10	6_23_11	8_45_10	8_45_11	8_6_9
9_01_4	9_01_5	9_23_4	9_23_5	11_45_4	11_45_5	11_6_4
9_01_6	9_01_7	9_23_6	9_23_7	11_45_6	11_45_7	11_6_5
10_01_0	10_01_1	10_23_0	10_23_1	12_45_0	12_45_1	12_6_0
10_01_2	10_01_3	10_23_2	10_23_3	12_45_2	12_45_3	12_6_1
10_01_8	10_01_9	10_23_8	10_23_9	12_45_8	12_45_9	12_6_8
10_01_10	10_01_11	10_23_10	10_23_11	12_45_10	12_45_11	12_6_9

Tables show the MIP/ISO bit assignment into cables for both sides of CMS.
Horizontal: ETA values between -6….+6
Vertical: PHI values 0….17 (20° units)
Syntax: **CableNr_EtaValues_BitNumberOnCable(starting with zero)**

6.2 Mapping GCT cables to PSB channels and backplane signals

PSB board ➔ BACKPLANE ➔ GMT

CH7_6: bit 31: CON1_19e – bit0: CON2_12d bit31-16 ➔ MEM7, bit15-0 ➔ MEM6
CH5_4: bit 31: CON2_15e – bit0: CON3_5d bit31-16 ➔ MEM5, bit15-0 ➔ MEM4
CH3_2: bit 31: CON4_1e – bit0: CON4_16d bit31-16 ➔ MEM3, bit15-0 ➔ MEM2
CH1_0: bit 31: CON5_1e – bit0: CON5_16d bit31-16 ➔ MEM1, bit15-0 ➔ MEM0

PSB in SLOT19: ➔ GCT Cables 1,2,3,4

blue number = bit# in PSB. The empty PSB bits are not connected to the GMT board.
Red MQ bits are sent parallel, to the barrel as well to the forward GMT Assignment logic

GCT cable 4 ➔ ch6_7

MEM7 ➔ bit31-16, MEM6 ➔ bit15-0

30	31
MQB4_45_8	MQB4_45_10
MQB4_45_4	MQB4_45_6
MQB4_45_0	MQB4_45_2
22	23
MQF4_6_8	MQF4_6_9
MQF4_6_4	MQF4_6_5
MQF4_6_0	MQF4_6_1
14	15
12	13
MQF4_45_10	MQF4_45_11
MQF4_45_8	MQF4_45_9
MQF4_45_6	MQF4_45_7
MQF4_45_4	MQF4_45_5
MQF4_45_2	MQF4_45_3
MQF4_45_0	MQF4_45_1

GCT cable 3 ➔ ch4_5

MEM5 ➔ bit31-16, MEM4 ➔ bit15-0

30	31
MQB3_45_8	MQB3_45_10
MQB3_45_4	MQB3_45_6
MQB3_45_0	MQB3_45_2

22	23
MQF3_6_8	MQF3_6_9
MQF3_6_4	MQF3_6_5
MQF3_6_0	MQF3_6_1
14	15
12	13
MQF3_45_10	MQF3_45_11
MQF3_45_8	MQF3_45_9
MQF3_45_6	MQF3_45_7
MQF3_45_4	MQF3_45_5
MQF3_45_2	MQF3_45_3
MQF3_45_0	MQF3_45_1

GCT cable 2 ➔ ch2_3

MEM3 ➔ bit31-16, MEM2 ➔ bit15-0

30	31
28	29
MQB2_23_10	MQB2_23_11
MQB2_23_8	MQB2_23_9
MQB2_23_6	MQB2_23_7
MQB2_23_4	MQB2_23_5
MQB2_23_2	MQB2_23_3
MQB2_23_0	MQB2_23_1

14	15
12	13
MQB2_01_10	MQB2_01_11
MQB2_01_8	MQB2_01_9
MQB2_01_6	MQB2_01_7
MQB2_01_4	MQB2_01_5
MQB2_01_2	MQB2_01_3
MQB2_01_0	MQB2_01_1

GCT cable 1 ➔ ch0_1

MEM1 ➔ bit31-16, MEM0 ➔ bit15-0

30	31
28	29
MQB1_23_10	MQB1_23_11
MQB1_23_8	MQB1_23_9
MQB1_23_6	MQB1_23_7
MQB1_23_4	MQB1_23_5
MQB1_23_2	MQB1_23_3
MQB1_23_0	MQB1_23_1
14	15
12	13
MQB1_01_10	MQB1_01_11
MQB1_01_8	MQB1_01_9
MQB1_01_6	MQB1_01_7
MQB1_01_4	MQB1_01_5
MQB1_01_2	MQB1_01_3
MQB1_01_0	MQB1_01_1

PSB in SLOT20:

GCT cable 8 ➔ ch6_7

MEM7 ➔ bit31-16, MEM6 ➔ bit15-0

30	31
MQB8_45_8	MQB8_45_10
MQB8_45_4	MQB8_45_6
MQB8_45_0	MQB8_45_2
22	23
MQF8_6_8	MQF8_6_9
MQF8_6_4	MQF8_6_5
MQF8_6_0	MQF8_6_1
14	15
12	13
MQF8_45_10	MQF8_45_11
MQF8_45_8	MQF8_45_9
MQF8_45_6	MQF8_45_7
MQF8_45_4	MQF8_45_5
MQF8_45_2	MQF8_45_3
MQF8_45_0	MQF8_45_1

GCT cable 7 ➔ ch4_5

MEM5 ➔ bit31-16, MEM4 ➔ bit15-0

30	31
MQB7_45_8	MQB7_45_10
MQB7_45_4	MQB7_45_6
MQB7_45_0	MQB7_45_2
22	23
MQF7_6_8	MQF7_6_9
MQF7_6_4	MQF7_6_5
MQF7_6_0	MQF7_6_1
14	15
12	13

MQF7_45_10	MQF7_45_11
MQF7_45_8	MQF7_45_9
MQF7_45_6	MQF7_45_7
MQF7_45_4	MQF7_45_5
MQF7_45_2	MQF7_45_3
MQF7_45_0	MQF7_45_1

GCT cable 6 ➔ ch2_3

MEM3 ➔ bit31-16, MEM2 ➔ bit15-0

30	31
28	29
MQB6_23_10	MQB6_23_11
MQB6_23_8	MQB6_23_9
MQB6_23_6	MQB6_23_7
MQB6_23_4	MQB6_23_5
MQB6_23_2	MQB6_23_3
MQB6_23_0	MQB6_23_1
14	15
12	13
MQB6_01_10	MQB6_01_11
MQB6_01_8	MQB6_01_9
MQB6_01_6	MQB6_01_7
MQB6_01_4	MQB6_01_5
MQB6_01_2	MQB6_01_3

MQB6_01_0	MQB6_01_1

GCT cable 5 ➔ ch0_1

MEM1 ➔ bit31-16, MEM0 ➔ bit15-0

30	31
28	29
MQB5_23_10	MQB5_23_11
MQB5_23_8	MQB5_23_9
MQB5_23_6	MQB5_23_7
MQB5_23_4	MQB5_23_5
MQB5_23_2	MQB5_23_3
MQB5_23_0	MQB5_23_1
14	15
12	13
MQB5_01_10	MQB5_01_11
MQB5_01_8	MQB5_01_9
MQB5_01_6	MQB5_01_7
MQB5_01_4	MQB5_01_5
MQB5_01_2	MQB5_01_3
MQB5_01_0	MQB5_01_1

The empty PSB bits are not connected to the GMT board.

PSB in SLOT21:

GCT cable 12 ➔ ch6_7

MEM7 ➔ bit31-16, MEM6 ➔ bit15-0

30	31
MQB12_45_8	MQB12_45_10
MQB12_45_4	MQB12_45_6
MQB12_45_0	MQB12_45_2
22	23
MQF12_6_8	MQF12_6_9
MQF12_6_4	MQF12_6_5
MQF12_6_0	MQF12_6_1
14	15
12	13
MQF12_45_10	MQF12_45_11
MQF12_45_8	MQF12_45_9
MQF12_45_6	MQF12_45_7
MQF12_45_4	MQF12_45_5
MQF12_45_2	MQF12_45_3
MQF12_45_0	MQF12_45_1

GCT cable 11 ➔ ch4-5,

MEM5 ➔ bit31-16, MEM4 ➔ bit15-0

30	31
MQB11_45_8	MQB11_45_10
MQB11_45_4	MQB11_45_6
MQB11_45_0	MQB11_45_2
22	23
MQF11_6_8	MQF11_6_9
MQF11_6_4	MQF11_6_5
MQF11_6_0	MQF11_6_1
14	15
12	13
MQF11_45_10	MQF11_45_11
MQF11_45_8	MQF11_45_9
MQF11_45_6	MQF11_45_7
MQF11_45_4	MQF11_45_5
MQF11_45_2	MQF11_45_3
MQF11_45_0	MQF11_45_1

GCT cable 10 ➔ ch2_3

MEM3 ➔ bit31-16, MEM2 ➔ bit15-0

30	31
28	29
MQB10_23_10	MQB10_23_11
MQB10_23_8	MQB10_23_9
MQB10_23_6	MQB10_23_7
MQB10_23_4	MQB10_23_5
MQB10_23_2	MQB10_23_3
MQB10_23_0	MQB10_23_1
14	15
12	13
MQB10_01_10	MQB10_01_11
MQB10_01_8	MQB10_01_9
MQB10_01_6	MQB10_01_7
MQB10_01_4	MQB10_01_5
MQB10_01_2	MQB10_01_3

MQB10_01_0	MQB10_01_1

GCT cable 9 ➔ ch0_1

MEM1 ➔ bit31-16, MEM0 ➔ bit15-0

30	31
28	29
MQB9_23_10	MQB9_23_11
MQB9_23_8	MQB9_23_9
MQB9_23_6	MQB9_23_7
MQB9_23_4	MQB9_23_5
MQB9_23_2	MQB9_23_3
MQB9_23_0	MQB9_23_1
14	15
12	13
MQB9_01_10	MQB9_01_11
MQB9_01_8	MQB9_01_9
MQB9_01_6	MQB9_01_7
MQB9_01_4	MQB9_01_5
MQB9_01_2	MQB9_01_3
MQB9_01_0	MQB9_01_1

The empty PSB bits are not connected to the GMT board.

6.3 AUF chip mapping to internal MQfwd bus in

Copied from vhdl code

-- mapping of MIP/QUIET bits as defined on cables and backplane

-- copied from MIPISObits.xls

```
MQfwd(0)  <= MQF3_6_0;
MQfwd(1)  <= MQF3_6_1;
MQfwd(2)  <= MQF3_6_8;
MQfwd(3)  <= MQF3_6_9;
MQfwd(4)  <= MQF4_6_4;
MQfwd(5)  <= MQF4_6_5;
MQfwd(6)  <= MQF7_6_0;
MQfwd(7)  <= MQF7_6_1;
MQfwd(8)  <= MQF7_6_8;
MQfwd(9)  <= MQF7_6_9;
MQfwd(10) <= MQF8_6_4;
MQfwd(11) <= MQF8_6_5;
MQfwd(12) <= MQF11_6_0;
MQfwd(13) <= MQF11_6_1;
MQfwd(14) <= MQF11_6_8;
MQfwd(15) <= MQF11_6_9;
MQfwd(16) <= MQF12_6_4;
MQfwd(17) <= MQF12_6_5;
MQfwd(18) <= MQF3_45_1;
MQfwd(19) <= MQF3_45_3;
MQfwd(20) <= MQF3_45_9;
MQfwd(21) <= MQF3_45_11;
MQfwd(22) <= MQF4_45_5;
MQfwd(23) <= MQF4_45_7;
MQfwd(24) <= MQF7_45_1;
MQfwd(25) <= MQF7_45_3;
MQfwd(26) <= MQF7_45_9;
MQfwd(27) <= MQF7_45_11;
MQfwd(28) <= MQF8_45_5;
MQfwd(29) <= MQF8_45_7;
MQfwd(30) <= MQF11_45_1;
MQfwd(31) <= MQF11_45_3;
MQfwd(32) <= MQF11_45_9;
MQfwd(33) <= MQF11_45_11;
MQfwd(34) <= MQF12_45_5;
MQfwd(35) <= MQF12_45_7;
MQfwd(36) <= MQF3_45_0;
MQfwd(37) <= MQF3_45_2;
MQfwd(38) <= MQF3_45_8;
MQfwd(39) <= MQF3_45_10;
MQfwd(40) <= MQF4_45_4;
MQfwd(41) <= MQF4_45_6;
MQfwd(42) <= MQF7_45_0;
MQfwd(43) <= MQF7_45_2;
MQfwd(44) <= MQF7_45_8;
MQfwd(45) <= MQF7_45_10;
MQfwd(46) <= MQF8_45_4;
MQfwd(47) <= MQF8_45_6;
MQfwd(48) <= MQF11_45_0;
MQfwd(49) <= MQF11_45_2;
MQfwd(50) <= MQF11_45_8;
MQfwd(51) <= MQF11_45_10;
MQfwd(52) <= MQF12_45_4;
MQfwd(53) <= MQF12_45_6;
MQfwd(54) <= MQF1_23_1;
MQfwd(55) <= MQF1_23_3;
MQfwd(56) <= MQF1_23_9;
MQfwd(57) <= MQF1_23_11;
MQfwd(58) <= MQF2_23_5;
MQfwd(59) <= MQF2_23_7;
MQfwd(60) <= MQF5_23_1;
MQfwd(61) <= MQF5_23_3;
MQfwd(62) <= MQF5_23_9;
MQfwd(63) <= MQF5_23_11;
MQfwd(64) <= MQF6_23_5;
MQfwd(65) <= MQF6_23_7;
MQfwd(66) <= MQF9_23_1;
MQfwd(67) <= MQF9_23_3;
MQfwd(68) <= MQF9_23_9;
MQfwd(69) <= MQF9_23_11;
MQfwd(70) <= MQF10_23_5;
MQfwd(71) <= MQF10_23_7;
MQfwd(72) <= MQF1_23_0;
MQfwd(73) <= MQF1_23_2;
MQfwd(74) <= MQF1_23_8;
MQfwd(75) <= MQF1_23_10;
MQfwd(76) <= MQF2_23_4;
MQfwd(77) <= MQF2_23_6;
MQfwd(78) <= MQF5_23_0;
MQfwd(79) <= MQF5_23_2;
MQfwd(80) <= MQF5_23_8;
MQfwd(81) <= MQF5_23_10;
MQfwd(82) <= MQF6_23_4;
MQfwd(83) <= MQF6_23_6;
```

```
MQfwd(84)  <= MQF9_23_0;
MQfwd(85)  <= MQF9_23_2;
MQfwd(86)  <= MQF9_23_8;
MQfwd(87)  <= MQF9_23_10;
MQfwd(88)  <= MQF10_23_4;
MQfwd(89)  <= MQF10_23_6;
MQfwd(90)  <= MQF1_23_4;
MQfwd(91)  <= MQF1_23_6;
MQfwd(92)  <= MQF2_23_0;
MQfwd(93)  <= MQF2_23_2;
MQfwd(94)  <= MQF2_23_8;
MQfwd(95)  <= MQF2_23_10;
MQfwd(96)  <= MQF5_23_4;
MQfwd(97)  <= MQF5_23_6;
MQfwd(98)  <= MQF6_23_0;
MQfwd(99)  <= MQF6_23_2;
MQfwd(100) <= MQF6_23_8;
MQfwd(101) <= MQF6_23_10;
MQfwd(102) <= MQF9_23_4;
MQfwd(103) <= MQF9_23_6;
MQfwd(104) <= MQF10_23_0;
MQfwd(105) <= MQF10_23_2;
MQfwd(106) <= MQF10_23_8;
MQfwd(107) <= MQF10_23_10;
MQfwd(108) <= MQF1_23_5;
MQfwd(109) <= MQF1_23_7;
MQfwd(110) <= MQF2_23_1;
MQfwd(111) <= MQF2_23_3;
MQfwd(112) <= MQF2_23_9;
MQfwd(113) <= MQF2_23_11;
MQfwd(114) <= MQF5_23_5;
MQfwd(115) <= MQF5_23_7;
MQfwd(116) <= MQF6_23_1;
MQfwd(117) <= MQF6_23_3;
MQfwd(118) <= MQF6_23_9;
MQfwd(119) <= MQF6_23_11;
MQfwd(120) <= MQF9_23_5;
MQfwd(121) <= MQF9_23_7;
MQfwd(122) <= MQF10_23_1;
MQfwd(123) <= MQF10_23_3;
MQfwd(124) <= MQF10_23_9;
MQfwd(125) <= MQF10_23_11;
MQfwd(126) <= MQF3_45_4;
MQfwd(127) <= MQF3_45_6;
MQfwd(128) <= MQF4_45_0;
MQfwd(129) <= MQF4_45_2;
MQfwd(130) <= MQF4_45_8;
MQfwd(131) <= MQF4_45_10;
MQfwd(132) <= MQF7_45_4;
MQfwd(133) <= MQF7_45_6;
MQfwd(134) <= MQF8_45_0;
MQfwd(135) <= MQF8_45_2;
MQfwd(136) <= MQF8_45_8;
MQfwd(137) <= MQF8_45_10;
MQfwd(138) <= MQF11_45_4;
MQfwd(139) <= MQF11_45_6;
MQfwd(140) <= MQF12_45_0;
MQfwd(141) <= MQF12_45_2;
MQfwd(142) <= MQF12_45_8;
MQfwd(143) <= MQF12_45_10;
MQfwd(144) <= MQF3_45_5;
MQfwd(145) <= MQF3_45_7;
MQfwd(146) <= MQF4_45_1;
MQfwd(147) <= MQF4_45_3;
MQfwd(148) <= MQF4_45_9;
MQfwd(149) <= MQF4_45_11;
MQfwd(150) <= MQF7_45_5;
MQfwd(151) <= MQF7_45_7;
MQfwd(152) <= MQF8_45_1;
MQfwd(153) <= MQF8_45_3;
MQfwd(154) <= MQF8_45_9;
MQfwd(155) <= MQF8_45_11;
MQfwd(156) <= MQF11_45_5;
MQfwd(157) <= MQF11_45_7;
MQfwd(158) <= MQF12_45_1;
MQfwd(159) <= MQF12_45_3;
MQfwd(160) <= MQF12_45_9;
MQfwd(161) <= MQF12_45_11;
MQfwd(162) <= MQF3_6_4;
MQfwd(163) <= MQF3_6_5;
MQfwd(164) <= MQF4_6_0;
MQfwd(165) <= MQF4_6_1;
MQfwd(166) <= MQF4_6_8;
MQfwd(167) <=    MQF4_6_9   ;
MQfwd(168) <=    MQF7_6_4   ;
MQfwd(169) <=    MQF7_6_5   ;
MQfwd(170) <=    MQF8_6_0   ;
MQfwd(171) <=    MQF8_6_1   ;
MQfwd(172) <=    MQF8_6_8   ;
MQfwd(173) <=    MQF8_6_9   ;
MQfwd(174) <=    MQF11_6_4  ;
MQfwd(175) <=    MQF11_6_5  ;
MQfwd(176) <=    MQF12_6_0  ;
MQfwd(177) <=    MQF12_6_1  ;
MQfwd(178) <=    MQF12_6_8  ;
MQfwd(179) <=    MQF12_6_9
```

6.4 AU Barrel Chip mapping

```
MQbrl(0)  <= MQB3_45_0;
MQbrl(1)  <= MQB3_45_2;
MQbrl(2)  <= MQB3_45_8;
MQbrl(3)  <= MQB3_45_10;
MQbrl(4)  <= MQB4_45_4;
MQbrl(5)  <= MQB4_45_6;
MQbrl(6)  <= MQB7_45_0;
MQbrl(7)  <= MQB7_45_2;
MQbrl(8)  <= MQB7_45_8;
MQbrl(9)  <= MQB7_45_10;
MQbrl(10) <= MQB8_45_4;
MQbrl(11) <= MQB8_45_6;
MQbrl(12) <= MQB11_45_0;
MQbrl(13) <= MQB11_45_2;
MQbrl(14) <= MQB11_45_8;
MQbrl(15) <= MQB11_45_10;
MQbrl(16) <= MQB12_45_4;
MQbrl(17) <= MQB12_45_6;
MQbrl(18) <= MQB1_23_1;
MQbrl(19) <= MQB1_23_3;
MQbrl(20) <= MQB1_23_9;
MQbrl(21) <= MQB1_23_11;
MQbrl(22) <= MQB2_23_5;
MQbrl(23) <= MQB2_23_7;
MQbrl(24) <= MQB5_23_1;
MQbrl(25) <= MQB5_23_3;
MQbrl(26) <= MQB5_23_9;
MQbrl(27) <= MQB5_23_11;
MQbrl(28) <= MQB6_23_5;
MQbrl(29) <= MQB6_23_7;
MQbrl(30) <= MQB9_23_1;
MQbrl(31) <= MQB9_23_3;
MQbrl(32) <= MQB9_23_9;
MQbrl(33) <= MQB9_23_11;
MQbrl(34) <= MQB10_23_5;
MQbrl(35) <= MQB10_23_7;
MQbrl(36) <= MQB1_23_0;
MQbrl(37) <= MQB1_23_2;
MQbrl(38) <= MQB1_23_8;
MQbrl(39) <= MQB1_23_10;
MQbrl(40) <= MQB2_23_4;
MQbrl(41) <= MQB2_23_6;
MQbrl(42) <= MQB5_23_0;
MQbrl(43) <= MQB5_23_2;
MQbrl(44) <= MQB5_23_8;
MQbrl(45) <= MQB5_23_10;
MQbrl(46) <= MQB6_23_4;
MQbrl(47) <= MQB6_23_6;
MQbrl(48) <= MQB9_23_0;
MQbrl(49) <= MQB9_23_2;
MQbrl(50) <= MQB9_23_8;
MQbrl(51) <= MQB9_23_10;
MQbrl(52) <= MQB10_23_4;
MQbrl(53) <= MQB10_23_6;
MQbrl(54) <= MQB1_01_1;
MQbrl(55) <= MQB1_01_3;
MQbrl(56) <= MQB1_01_9;
MQbrl(57) <= MQB1_01_11;
MQbrl(58) <= MQB2_01_5;
MQbrl(59) <= MQB2_01_7;
MQbrl(60) <= MQB5_01_1;
MQbrl(61) <= MQB5_01_3;
MQbrl(62) <= MQB5_01_9;
MQbrl(63) <= MQB5_01_11;
MQbrl(64) <= MQB6_01_5;
MQbrl(65) <= MQB6_01_7;
MQbrl(66) <= MQB9_01_1;
MQbrl(67) <= MQB9_01_3;
MQbrl(68) <= MQB9_01_9;
MQbrl(69) <= MQB9_01_11;
MQbrl(70) <= MQB10_01_5;
MQbrl(71) <= MQB10_01_7;
MQbrl(72) <= MQB1_01_0;
MQbrl(73) <= MQB1_01_2;
MQbrl(74) <= MQB1_01_8;
MQbrl(75) <= MQB1_01_10;
MQbrl(76) <= MQB2_01_4;
MQbrl(77) <= MQB2_01_6;
MQbrl(78) <= MQB5_01_0;
MQbrl(79) <= MQB5_01_2;
MQbrl(80) <= MQB5_01_8;
MQbrl(81) <= MQB5_01_10;
MQbrl(82) <= MQB6_01_4;
MQbrl(83) <= MQB6_01_6;
MQbrl(84) <= MQB9_01_0;
MQbrl(85) <= MQB9_01_2;
MQbrl(86) <= MQB9_01_8;
MQbrl(87) <= MQB9_01_10;
MQbrl(88) <= MQB10_01_4;
MQbrl(89) <= MQB10_01_6;
```

```
MQbrl(90)  <= MQB1_01_4;
MQbrl(91)  <= MQB1_01_6;
MQbrl(92)  <= MQB2_01_0;
MQbrl(93)  <= MQB2_01_2;
MQbrl(94)  <= MQB2_01_8;
MQbrl(95)  <= MQB2_01_10;
MQbrl(96)  <= MQB5_01_4;
MQbrl(97)  <= MQB5_01_6;
MQbrl(98)  <= MQB6_01_0;
MQbrl(99)  <= MQB6_01_2;
MQbrl(100) <= MQB6_01_8;
MQbrl(101) <= MQB6_01_10;
MQbrl(102) <= MQB9_01_4;
MQbrl(103) <= MQB9_01_6;
MQbrl(104) <= MQB10_01_0;
MQbrl(105) <= MQB10_01_2;
MQbrl(106) <= MQB10_01_8;
MQbrl(107) <= MQB10_01_10;
MQbrl(108) <= MQB1_01_5;
MQbrl(109) <= MQB1_01_7;
MQbrl(110) <= MQB2_01_1;
MQbrl(111) <= MQB2_01_3;
MQbrl(112) <= MQB2_01_9;
MQbrl(113) <= MQB2_01_11;
MQbrl(114) <= MQB5_01_5;
MQbrl(115) <= MQB5_01_7;
MQbrl(116) <= MQB6_01_1;
MQbrl(117) <= MQB6_01_3;
MQbrl(118) <= MQB6_01_9;
MQbrl(119) <= MQB6_01_11;
MQbrl(120) <= MQB9_01_5;
MQbrl(121) <= MQB9_01_7;
MQbrl(122) <= MQB10_01_1;
MQbrl(123) <= MQB10_01_3;
MQbrl(124) <= MQB10_01_9;
MQbrl(125) <= MQB10_01_11;
MQbrl(126) <= MQB1_23_4;
MQbrl(127) <= MQB1_23_6;
MQbrl(128) <= MQB2_23_0;
MQbrl(129) <= MQB2_23_2;
MQbrl(130) <= MQB2_23_8;
MQbrl(131) <= MQB2_23_10;
MQbrl(132) <= MQB5_23_4;
MQbrl(133) <= MQB5_23_6;
MQbrl(134) <= MQB6_23_0;
MQbrl(135) <= MQB6_23_2;
MQbrl(136) <= MQB6_23_8;
MQbrl(137) <= MQB6_23_10;
MQbrl(138) <= MQB9_23_4;
MQbrl(139) <= MQB9_23_6;
MQbrl(140) <= MQB10_23_0;
MQbrl(141) <= MQB10_23_2;
MQbrl(142) <= MQB10_23_8;
MQbrl(143) <= MQB10_23_10;
MQbrl(144) <= MQB1_23_5;
MQbrl(145) <= MQB1_23_7;
MQbrl(146) <= MQB2_23_1;
MQbrl(147) <= MQB2_23_3;
MQbrl(148) <= MQB2_23_9;
MQbrl(149) <= MQB2_23_11;
MQbrl(150) <= MQB5_23_5;
MQbrl(151) <= MQB5_23_7;
MQbrl(152) <= MQB6_23_1;
MQbrl(153) <= MQB6_23_3;
MQbrl(154) <= MQB6_23_9;
MQbrl(155) <= MQB6_23_11;
MQbrl(156) <= MQB9_23_5;
MQbrl(157) <= MQB9_23_7;
MQbrl(158) <= MQB10_23_1;
MQbrl(159) <= MQB10_23_3;
MQbrl(160) <= MQB10_23_9;
MQbrl(161) <= MQB10_23_11;
MQbrl(162) <= MQB3_45_4;
MQbrl(163) <= MQB3_45_6;
MQbrl(164) <= MQB4_45_0;
MQbrl(165) <= MQB4_45_2;
MQbrl(166) <= MQB4_45_8;
MQbrl(167) <= MQB4_45_10;
MQbrl(168) <= MQB7_45_4;
MQbrl(169) <= MQB7_45_6;
MQbrl(170) <= MQB8_45_0;
MQbrl(171) <= MQB8_45_2;
MQbrl(172) <= MQB8_45_8;
MQbrl(173) <= MQB8_45_10;
MQbrl(174) <= MQB11_45_4;
MQbrl(175) <= MQB11_45_6;
MQbrl(176) <= MQB12_45_0;
MQbrl(177) <= MQB12_45_2;
MQbrl(178) <= MQB12_45_8;
MQbrl(179) <= MQB12_45_10;
```