28.5 Advanced Routing Unit (ARU)

28.5.1 Overview

The Advanced Routing Unit (ARU) is a flexible infrastructure component for transferring 53 bit wide data (five control bits and two 24 bit values) between several sub-modules of the GTM core in a configurable manner.

Since the concept of the ARU has already been described in the paragraph “ARU routing concept”, this section only describes additional ARU features that can be used by the software for configuring and debugging ARU related data streams.Also the definition of 'streams' and 'channels' in the ARU context is done in “ARU routing concept”.

The principle of ARU data routing is described in “ARU Write Address Overview”. In the real GTM implementation the ARU serves in parallel per clock period two individual data destinations, one destination at port ARU-0 and at port ARU-1. Both ARU ports ARU-0 and ARU-1 are running by default in parallel but can be configured in dynamic routing mode (see below) to run in an individual mode.

As already defined in the “ARU routing concept”, the ARU read ID is the address of the data source that is configured in the data destination module. These ARU read ID's are selected by the individual counter of ARU ports ARU-0 and ARU-1.

Via the ARU ports ARU-0 and ARU-1 with each ARU read ID two independent GTM sub-modules are addressed and served. The combination of ARU port (ARU-0 or ARU-1) and the ARU read ID addresses one ARU wdata source (i.e. the ARU write port of a GTM sub-module).

The assignment of ARU write ports of GTM sub-modules to the ARU ports ARU-0 and ARU-1 and the ARU read ID's is device depending and can be found in the corresponding sub-chapter.

28.5.2 Special Data Sources

Besides the addresses of the sub-module related data sources as described in product specific appendix, the ARU provides two special data sources that can be used for the configuration of data streams. These data sources are defined as follows:

Address 0x1FF: Data source that provides always a 53 bit data word with zeros. A read access to this memory location will never block a requesting data destination.

Address 0x1FE: Data source that never provides a data word. A read access to this memory location will always block a requesting data destination. This is the reset value of the read registers inside the data destinations.

Address 0x000: This address is reserved and can be used to bring data through the ARU registers **ARU_DATA_H** and **ARU_DATA_L** into the system by writing the write address 0x000 into the **ARU_ACCESS** register. This means that software test data can be brought into the GTM by the CPU.

28.5.3 ARU Access via AEI

Besides the data transfer between the connected sub-modules, there are two possibilities to access ARU data via the AEI.

28.5.3.1 Default ARU Access

The default ARU access incorporates the registers **ARU_ACCESS**, which is used for initiation of a read or write request and the registers **ARU_DATA_H** and **ARU_DATA_L** that provide the ARU data word to be transferred.

The status of a read or write transfer can be determined by polling specific bits in register **ARU_ACCESS**. Furthermore the *acc_ack* bit in the interrupt notify register is set after the read or write access is performed to avoid data loss e.g. on access cancelation.

A pending read or write request may also be canceled by clearing the associated bit.

In the case of a read request, the AEI access behaves as a read request initiated by a data destination of a module. The read request is served by the ARU immediately when no other destination has a pending read request. This means, that an AEI read access does not take part in the scheduling of the destination channels and that the time between two consecutive read accesses is not limited by the round trip time.

On the other hand, the AEI access has the lowest priority behind the ARU scheduler that serves the destination channels. Thus, in worst case, the read request is served after one round trip of the ARU, when all destination channels would request data at the same point in time.

In the case of the write request, the ARU provides the write data at the address defined by the ADDR bit field inside the **ARU_ACCESS** register.

To avoid data loss, the reserved ARU address 0x0 has to be used to bring data into the system. Otherwise, in case the address specified inside the ADDR bit field is defined for another sub-module that acts as a source at the ARU data loss may occur and no deterministic behavior is guaranteed.

This is because the regular source sub-module is not aware that its address is used by the ARU itself to provide data to a destination.

It is guaranteed that the ARU write data is send to the destination in case of both modules want to provide data at the same time.

Configuring both read and write request bits results in a read request, if the write request bit inside the register isn't already set. The read request bit will be set but not the write request bit. The following table describes the important cases of the bit 12 (RREQ) and bit 13 (WREQ) of the **ARU_ACCESS** register:

Table 12 WREQ and RREQ in ARU_ACCESS register

AEI write access: aei_wdata (13:12)	actual value of ARU_ACCESS(13:12)	next value of ARU_ACCESS(13:12)	comment
0 0	0 1	0 0	cancel read request
0 0	1 0	0 0	cancel write request
0 1	1 0	1 0	unchanged register
1 0	0 1	0 1	unchanged register
1 1	0 0	0 1	both read and write request results in a read request
1 1	1 0	1 0	as before but WREQ bit is already set -> unchanged register

28.5.3.2 Debug Access

The debug access mode enables to inspect routed data of configured data streams during runtime.

The ARU provides two independent debug channels, whereas each is configured by a dedicated ARU read address in register **ARU_DBG_ACCESS0** and **ARU_DBG_ACCESS1** respectively.

The registers **ARU_DBG_DATA0_H** and **ARU_DBG_DATA0_L** (**ARU_DBG_DATA1_H** and **ARU_DBG_DATA1_L**) provide read access to the latest data word that the corresponding data source sent through the ARU.

Any time when data is transferred through the ARU from a data source to the destination requesting the data the interrupt signal *ARU_NEW_DATA0*_IRQ (*ARU_NEW_DATA1*_IRQ) is raised.

For advanced debugging purposes, the interrupt signal can also be triggered by software using the register **ARU_IRQ_FORCINT**.