TNO report

FEL-98-A052

COTS Interface FCM

TNO Physics and Electronics
Laboratory

Oude Waalsdorperweg 63
PO Box 96864
2509 JG The Hague
The Netherlands

Phone +31 70 374 00 00
Fax +31 70 328 09 61

Date
July 1998

Author(s)
L.H.J. Bierens
C. van 't Wout
P.A.M. Stijnman

Sponsor	Royal Netherlands Airforce
Project officer	G.J. de Wilde
Affiliation	DMKLu/MXS

Classification	
Classified by	G.J. de Wilde
Classification date	3 July 1998
Title	Ongerubriceerd
Managementuittreksel	Ongerubriceerd
Abstract	Ongerubriceerd
Report text	Ongerubriceerd
Appendix	Ongerubriceerd

Copy no	12
No of copies	28
No of pages	41 (incl appendix, excl RDP & distribution list)
No of appendices	1

19990119 050

AQ F99-04-0654

Managementuittreksel

Titel	:	COTS Interface FCM
Auteur(s)	:	Dr.ir. L.H.J. Bierens, Ing. C. van 't Wout, Ing. P.A.M. Stijnman
Datum	:	juli 1998
Opdrachtnr.	:	A97KLu735
IWP-nr.	:	771
Rapportnr.	:	FEL-98-A052

Dit rapport beschrijft de specificaties van het resultaat van het project "COTS Interface FCM". Het doel van dit project was het specificeren, het ontwerpen en het realiseren van een interface tussen de high-speed real-time Synthetic Aperture Radar (SAR) systeem databus (RACEway) aan de ene kant, en de Fast Convolution Module (FCM) aan de andere kant.

Een Synthetic Aperture Radar (SAR) is een side-looking beeldvormende radar gemonteerd onder een bewegend platform zoals een vliegtuig of een satelliet. Met een SAR is het mogelijk om afbeeldingen van het aardoppervlak te genereren van bijna fotokwaliteit. Een SAR heeft all-weather en day-and-night capability, en is daardoor uitermate geschikt voor bijv. surveillance en reconnaissance taken, bijv. ingebouwd in een verkenningspod of een UAV.

Voor defensietoepassingen is het van belang dat SAR afbeeldingen tijdens de opname *in real-time* beschikbaar zijn. Het genereren van een SAR afbeelding vereist zeer veel specifieke rekencapaciteit, die op standaard computerplatforms, zoals PC's of workstations, niet beschikbaar is. TNO-FEL heeft een multi-purpose real-time SAR processor ontwikkeld die wel de vereiste rekencapaciteit bezit. Deze processor is gebaseerd op commercieel verkrijgbare Digital Signal Processing (DSP) boards, aangevuld met in-huis ontwikkelde dedicated hardware zoals de FCM. De FCM voert een van de kritische SAR processing stappen uit, nl. range compressie. Door het gebruik van de FCM in plaats van commercieel verkrijgbare DSP boards wordt op zijn minst een factor 3 aan omvang, vermogensverbruik en kosten bespaard op de benodigde hardware.

Het ontwerp van de COTS (Commercial-Off-The-Shelf) interface is generiek van opzet. In de toekomst kan het worden gebruikt om andere dedicated hardware subsystemen te integreren in de multi-purpose real-time SAR processing omgeving. Het projectresultaat, een hardware realisatie van de interface, is met succes toegepast bij de integratie van de FCM in de multi-purpose real-time SAR processor.

De COTS interface maakt de integratie mogelijk van COTS producten in high-performance signaalverwerkende systemen waarbij flexibele processing m.b.v. van DSP boards en "bulk processing" m.b.v. dedicated hardware op elegante wijze

samengaan. Met dit project heeft TNO-FEL laten zien dat deze "best-of-both-worlds" aanpak leidt tot een substantiële reductie in omvang, vermogensverbruik en kosten in complexe applicatie-specifieke signaalverwerkingssystemen zoals de multi-purpose real-time SAR processor.

De multi-purpose real-time SAR processor is voor de Krijgsmachtdelen beschikbaar voor experimentele inzet t.b.v. defensietoepassingen. Met de evaluatie hiervan kan een Krijgsmachtdeel haar behoeftestelling onderbouwen, en bovendien als "smart buyer" optreden naar aanbieders van end-to-end SAR systemen. Ook kan met behulp van deze real-time SAR processor een aangeboden SAR systeem relatief eenvoudig worden geëvalueerd op performance en specifieke gebruikerseisen van de Krijgsmacht. De specificatie van het SAR systeem kan vervolgens optimaal worden aangepast aan de behoeftestelling.

Contents

List of Abbreviations

ANSI	American National Standards Institute
CE	Compute Environment
COTS	Commercial Off The Shelf
DSP	Digital Signal Processing
FCM	Fast Convolution Module
FIFO	First In First Out memory
ILK	Interlink
I/O	Input-Output
O/B	On-Board
PCB	Printed Circuit Board
RASSP	Rapid Prototyping of Application Specific Signal Processors
RT-SAR	Real-Time SAR
SAR	Synthetic Aperture Radar
SATSAR	Satellite SAR
VITA	VMEbus International Trade Association
VME	Versa Modula Europa (an industry standard, high-end, open architecture computer system)

1.   Introduction

1.1   Background

Recently, TNO-FEL developed a multi-purpose real-time SAR (RT-SAR) processor. The processor environment was based on technology developed in the framework of two projects (see Section 1.4):
1) Quick-Look Processor;
2) Multi-purpose real-time SAR processing.

The objective of this processor is twofold:
*   to demonstrate new processing technologies developed for real-time on-board SAR processing, and evaluate them on flexibility, performance, image quality and costs, with respect to the application,
*   to evaluate advanced SAR applications, such as polarimetry, high-resolution SAR, satellite SAR (SATSAR) processing and interferometry, in a real-time SAR environment, and their impact on the processing technology.

The Fast Convolution Module (FCM) is an essential subsystem of the multi-purpose RT-SAR processor, developed in the framework of the project "Fast Convolution Module", see Section 1.4. It performs the range and / or azimuth compression steps as part of the complete SAR processing chain. Range and azimuth compression are referred to as "bulk-processing" [BIE97].

Bulk-processing applications can be implemented efficiently in dedicated hardware. The bulk-processing typically requires low flexibility and has high algorithmic regularity. Bulk-processing implementations can be optimized in dedicated hardware, whereas general purpose high performance hardware typically supports flexibility and a wide range of algorithms.

1.2   Objective of this Project

The objective of the project was the specification, the design and the realisation of an interface between the high-speed RT-SAR system data bus (the RACEway) at one side and the FCM at the other side.

The design of the COTS interface is generic. In the future, it can be used for other dedicated hardware subsystems as part of the multi-purpose RT-SAR processor environment.

Figure 1.1 shows a schematic block diagram of the COTS interface. It consists of three parts:
1) a first generic part: the RACEway interface,
2) a second generic part: the VME bus interface,
3) a subsystem dependent part: the FCM controller.

The VME bus interface is a low-speed data and control bus. For the FCM the high-speed data bus is sufficient. Therefore the VME bus interface is not realised in this project. In the future, however, the interface can be extended with the VME bus interface.
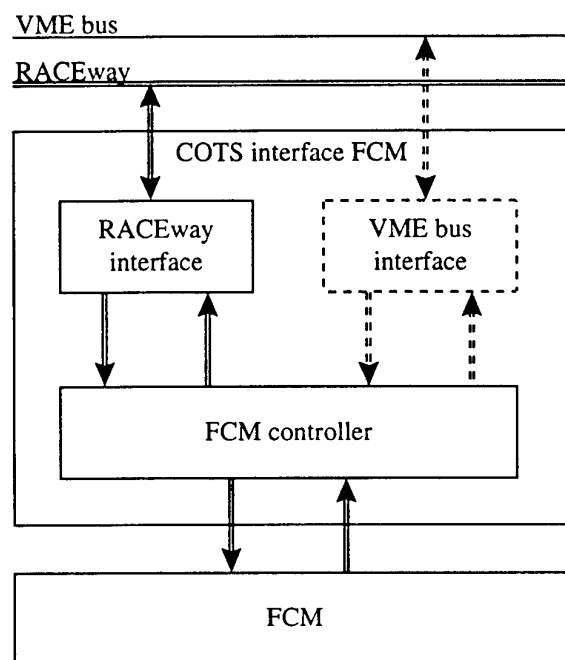
Figure 1.1: Schematic view of the COTS interface FCM.

1.3   Outline of this Report

This report describes the specifications of the COTS interface. The interface PCB is realised and tested, and integrated in the multi-purpose RT-SAR processor.

First, the system context is described in Section 2. The specification and the design of the RACEway interface are the subjects of Sections 3 and 4, respectively. The design of the FCM controller is the subject of Section 5. Appendix A gives the specification of the optional VME bus interface.

1.4   Related Projects

project title	funding	project code	TNO report no.
Testbedomgeving real-time SAR	KL	A94KL733	FEL-94-A292
Fast Convolution Module	Klu	A93KLu777	FEL-96-A307
Feasibility study real-time SATSAR	NIVR	NRT 2504 FE	in preparation
Quick-Look Processor	CO	A96D813	in preparation
Multi-purpose real-time SAR processor	NIVR	NRT 2701 FE	in preparation

2.   System Context

The overall requirements of the multi-purpose RT-SAR processor are the following:

- it should be an open "embedded system" architecture, that is, the developing engineers must have the ability to develop integrated parts of the architecture without substantial affecting the overall system configuration,
- it should have a wide acceptance in the relevant industrial sectors, and preferably support a broad variety of "company-independent" commercial-off-the-shelf components (COTS),
- it should be possible to implement less-critical processing elements in the SAR processing chain in programmable DSP hardware, and critical processing elements in dedicated hardware,
- the development environment should be robust and (as much as possible) user-friendly, and it must support heterogeneous, parallel DSP system development,
- generic input and output interfacing of the system should be available.

Based on an extensive market search TNO-FEL has selected the heterogeneous RACEway architecture [MCIO96], [MCDG95] of Mercury Computer Systems as the best candidate for its purpose. The concept of the architecture allows for an upgrade to, for example, real-time SATSAR processing. In Figure 2.1 the global architecture is shown as developed at TNO Physics and Electronics Laboratory.
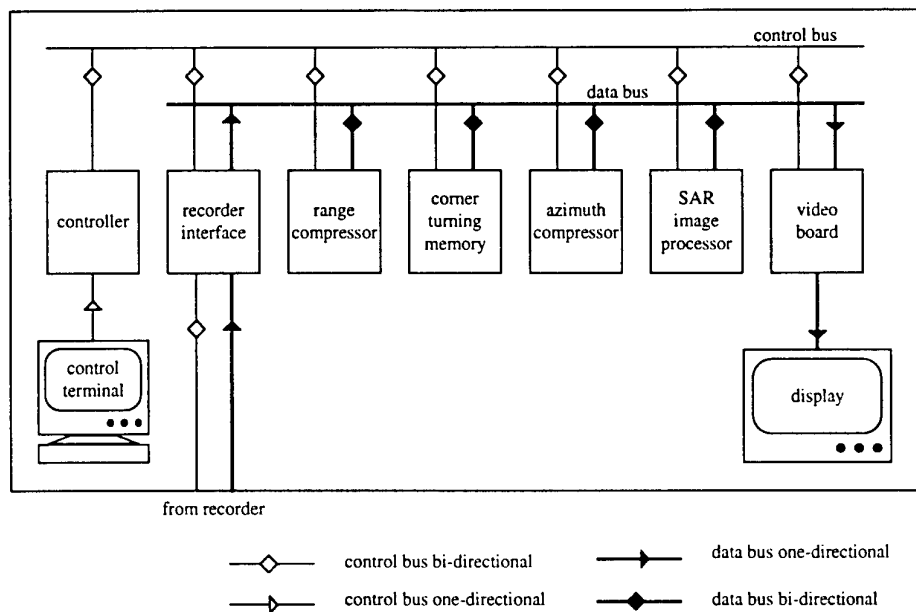
Figure 2.1:   System configuration of the multi-purpose RT-SAR processor.

The multi-puprose RT-SAR processing environment must allow an improvement of the design cycle of complex DSP systems, such as the SAR system, in terms of

time and costs. Modularity and flexibility are key words in the system design approach. The system environment uses:

a) standard VME bus format [VME82] with RACEway high-speed data bus extension [VITA94],

b) standard interfaces between subsystems and system bus,

c) flexible COTS front-end and back-end of the system.

This system is considered mandatory in emerging new technologies for advanced On-Board (O/B) RT-SAR applications, as is proposed in the Rapid prototyping of Application Specific Signal Processors (RASSP) program in the US [RASSP95]. However, unlike the RASSP program in the US, the multi-purpose RT-SAR processor is not only dedicated to airborne applications, but also to spaceborne applications.

3.   RACEway Interface Requirements

The RACEway interlink will provide in the RT-SAR system a high bandwidth data
path between PCBs. To let hardware subsystems make maximum use of the
RACEway interlink protocol, the following requirements are defined for the
interface between a hardware system and the RACEway interlink:

- it will minimise communication overhead for a hardware subsystem,
- the PCB area needed by the interface components will be minimised,
- where possible, the interface is built with off-the-shelf components,
- to make communication possible with custom PCBs, it is necessary to agree to
  the complete RACEway interlink specifications,
- the interface specification must give space to the various needs of the different
  hardware subsystems,
- data traffic must be possible in two directions, from a hardware subsystem to
  the RACEway interlink and vice versa.

3.1   RACEway interlink side of the interface

On the RT-SAR system the RACEway interlink can be implemented as follows.
The RT-SAR system is a 19" rack with some U6 slots. The upper connector (P1) is
used for the VME interface. The lower connector (P2) is used by the RACEway
interlink. The VME bus and the RACEway interlink can be implemented on the
same PCB, operations can take place simultaneously on both interfaces without
interaction. A RACEway interlink module can be placed on the back of four P2
connectors (Figure 3.1). One RACEway interlink module makes it possible to let
four PCBs communicate with each other and makes it possible to make a
connection to two neighbouring RACEway interlink modules. The RACEway
interlink can be increased by adding more RACEway interlink modules and by
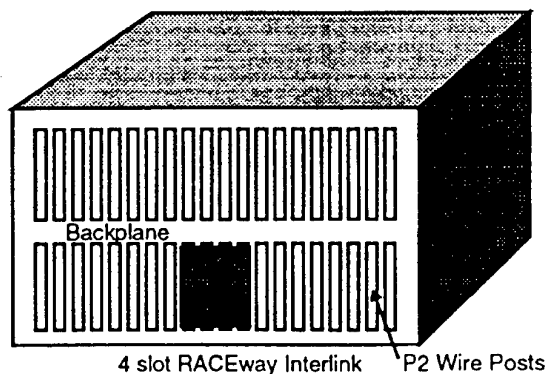connecting them together.

Backplane

4 slot RACEway Interlink   P2 Wire Posts

Figure 3.1:   Implementation of the RACEway interlink.

The RACEway interlink makes it possible to make a point-to-point connection from any node to any other node on the RACEway interlink. The data transfer has a bandwidth of 160Mbyte/sec. It is also possible to broadcast data. One node then writes data to multiple other nodes.

Multiple connections can be made simultaneously and each connection keeps a bandwidth of 160Mbyte/sec., e.g. node A can have a connection to node B while node C has a connection with node D. The total bandwidth will be 320Mbyte/sec for this example.

A connection between nodes can only be made by a master. The slave can be written to or read by the master when a connection is established. To establish a connection, the master adds a header to the data as is shown in Table 3.1.

Table 3.1:   *Data sequence.*

Route word[31:0]
Address word[31:0]
Data word 0[31:0]
Data word 0[31:0]
Data word 1[31:0]
Data word 1[31:0]
...

The header contains routing information to find a path through the RACEway interlink modules. When the header reaches the slave node, the slave acknowledges the connection, and a circuit through the RACEway interlink modules is established. The circuit through the modules is held open for the transaction behind the header. The master node is then able to write or read data through the established circuit.

3.1.1   Route word
The route word contains the information shown in Table 3.2.

Table 3.2:   *Route word format.*

31..5	4..3	2..1	0
Route / high order address field	Broadcast accept code	Routing priority	Broadcast / single mode

Route
A connection between two nodes is made using one or more RACEway interlink modules. The route word defines which RACEway interlink modules are used to make a connection. Figure 3.2 shows an example of this.
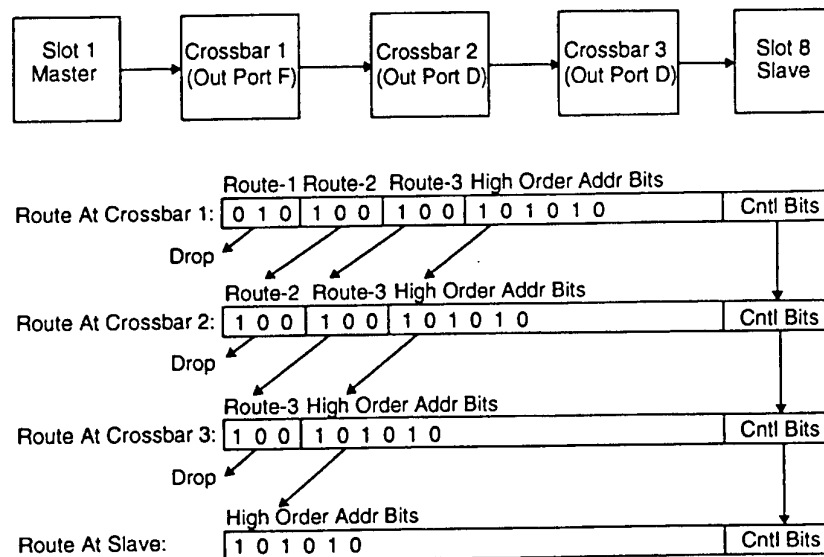
Figure 3.2:   Routing example.

The first RACEway interlink module uses only the first three bits of the routing word to establish the connection to the second RACEway interlink module. Before passing the route word to the next RACEway interlink module, the route word is shifted three places left. This process is repeated at the next RACEway interlink module until the route word reaches a node.

High order address

By passing RACEway interlink modules when making a connection, the route bits are shifted left three places each time a module is passed. At the destination, the remaining bits of the Route/Address field may be used by the slave as additional address bits. Conceptually, only 6 bits may be used for this purpose. In practice, more than 6 bits can be used to extend the address range.

Broadcast accept code

The route word contains a broadcast acceptance code. These bits are only used when data is broadcasted. Slaves may accept the arriving broadcast if the acceptance code matches the slave's acceptance key. A master may use this mechanism to select up to four different subpopulations of slaves for multicast operation.

The slave shall respond to the broadcast transaction whether or not the acceptance code matches the key.

Multicast operation may also be performed by using the broadcast route codes to route a broadcast to a subset of the nodes. A combination of the two methods may also be used.

Routing priority

Each transaction has a 2 bits routing priority. These bits make it possible to let high priority messages take precedence over low priority messages. The lower priority transaction will automatically be killed when a higher priority transaction needs a connection. When the higher priority transaction is finished, the lower priority transaction will automatically start back up where it left off.

Broadcast / single mode

The interpretation of the routing word depends on the setting of the broadcast/single mode bit. When in the single mode, an input is connected to only one output. When in the broadcast mode, an input is connected to multiple outputs. What connection is made exactly is device dependent, so for specific information the device specifications should be consulted.

3.1.2   Address word

The address word contains the information shown in Table 3.3.

Table 3.3:   Address word format.

31..28	27..3	2	1	0
Width / alignment	Address	Reserved	Read flag	Lock flag

Width / alignment

A RACEway slot may use the RACEway to access data that is 1 byte, 2 bytes, 4 bytes or 8 bytes wide. In addition, most alignments of 1-byte, 2-byte and 4-byte data are supported. The header contains a width/alignment word which gives information about the place of the interesting data when 1-byte, 2-byte or 4-byte data words are used. Block mode operation will only support 8-byte words.

Address

The address word gives place to a 25 bits address. If the data route crosses no more than 7 RACEway interlink modules, then the route word gives place to 6 additional address bits. Each data word has 64 bits, which gives a total addressing space of 2^{34} bytes.

Read flag

The read flag tells the direction of the data transaction. For read, read-modified-write and split read operations, the flag shall be set. For writes and broadcast operations, the flag shall be 0.

Lock flag

To prevent that a transaction can be killed by a higher priority transaction, the lock flag must be reset.

3.1.3   Data bus

The data path on each port of the RACEway interlink module is conceptually 8 bytes wide. Physically, the data path is 4 bytes wide, with logic automatically transferring 8 byte quantities over two consecutive cycles. This double cycling is transparent to software running on the master.

3.1.4   Physical layer specification

A RACEway connection has 32 data signals and 8 control signals. The signals between a RACEway node and a RACEway interlink module are the same signals as between RACEway interlink modules. The following signals are used for a RACEway connection:

Signal: RDCONIO.
Direction: Send by the slave and received by the master.
Function: Read control. A slave activates this signal when it wants to use the data bus for two cycles. For a master the read control is an input and when activated, the master shall tri-state the data bus for two cycles so read data can be driven. It is also used to indicate a read error.

Signal: RPLYIO.
Direction: Send by a slave and received by a master.
Function: Reply has several functions.
1.   The slave asserts reply when it has received a request in. By asserting reply, the slave indicates that it is ready to receive the address.
2.   The slave shall assert reply to indicate that it wants to split the read and it is ready to receive the return route.
3.   The slave asserts reply during read operations to indicate that it is driving data on the data bus.
4.   The slave asserts reply during write operations to indicate that it is ready to receive data. One reply pulse is sent for every word the slave is able to receive.

Signal: REQ_I.
Direction: From a RACEway interlink module to a node.
Function: Request in is received by a slave node when the RACEway interlink module is requesting control of the data bus. Request in is received by a master when a transaction must by killed. When certain conditions are met, the master node will stop its transaction and drop request out.

Signal: REQ_O.
Direction: From the interface to the RACEway interlink module.
Function: Request out is asserted by the master to request access to the data bus. It then stays asserted as long as the master is in control. It shall only be asserted if 'request in' was inactive during the previous period.

Signal: STROBIO.
Direction: For a master it is an output and for a slave it is an input.
Function: Strobe indicates that address or data is being send on the data bus.
             Strobe is sent by the master node after asserting REQ_O.

Signal: BIO[31:0].
Direction: From the interface to the RACEway and vice versa.
Function: Carries route, address and data between master and slave.

Signal: CLKI.
Direction: From the RACEway to the interface.
Function: Clock, provides the RACEway timing. High speed configurations will
             use a 40MHz clock and normal configurations use a 33MHz clock
             frequency.

Signal: RESETIO.
Direction: From the slave to the master (for the RIC-RINO chipset it is only an
             input).
Function: Reset. A master can reset the RACEway interlink module by asserting
             this signal and the RACEway interlink module is able to reset the
             internal logic of a slave with this signal.

Signal: SYNCI.
Direction: From the RACEway interlink to the interface.
Function: Sync provides control phase information to the RACEway interlink.

3.2   Subsystem side of the interface

There are some different subsystems which will make use of this side of the
interface. Each subsystem will have its own requirements for the interface.
Because of the various needs, it seems to be the best solution to define multiple
modules in the interface which can be left out when they are not required.

A distinction must be made between subsystems which need the ability to start a
communication and subsystems which will only communicate when other
subsystems ask for it. The first subsystems are called masters and the second
subsystems are called slaves. The masters need extra control logic for starting a
communication. This means for example totally different logic at the RACEway
interlink side of the interface and that at the subsystem side a route and an address
word must be supplied.

The interface must have the ability to read or write, no matter it is a master or a
slave. When only one way traffic is needed, the interface can be stripped. The

RACEway interlink deals with 64 bit data words, time multiplexed on a 32 bits (physical) data bus. Some extra logic is needed to undo the time multiplexing.

Multiple addressing methods are possible. The master of a transaction can supply an address to the slave at the start of each data burst. This address is 25 bits wide and can be extended to 31 bits when no more than 7 RACEway interlink modules are crossed to make the connection. This is only a start address and extra logic is needed to generate the following addresses.

When the bandwidth of the RACEway interlink is sufficient, the address could be multiplexed with the data. In this case, the master of a transaction generates data and address, multiplexes them on the data bus and sends them to the slave, where they are demultiplexed. This will only work for write actions.

There are some control lines needed between subsystem and interface. In Table 3.4 possible lines are listed for data transport from the RACEway to the subsystem.

Table 3.4:   *Control lines for data transport from the RACEway to the subsystem.*

Line	Function
Ready	The interface has valid data available.
Suspend	The subsystem is not able to receive new data.
Sync	A marker to separate succeeding sequences.
Clock	The subsystem clock, asynchronous to the RACEway clock.
Dir	With a combined read / write data bus a direction line is needed.

In Table 3.5 possible lines are listed data transport from the subsystem to the RACEway:

Table 3.5:   *Control lines for data transport from the subsystem to the RACEway.*

Line	Function
Valid	The subsystem has valid data available.
Full	The interface is not able to receive new data.
Sync	A marker to separate succeeding sequences.
Clock	The subsystem clock, asynchronous to the RACEway clock.

Control words from the master to the slave can be time-multiplexed on the data bus. A protocol about these things can be user defined and can be different for each connection. Control words that may be useful, are: length of the data burst and a burst identification number.

4. RACEway Interface Design

4.1 The RACEway Backplane

4.1.1 Routing a data transfer

Boards are interconnected by crossbar chips which are located on the backplane of the P2 connector. In our case, that they are located on an ILK8 module (Figure 4.1).
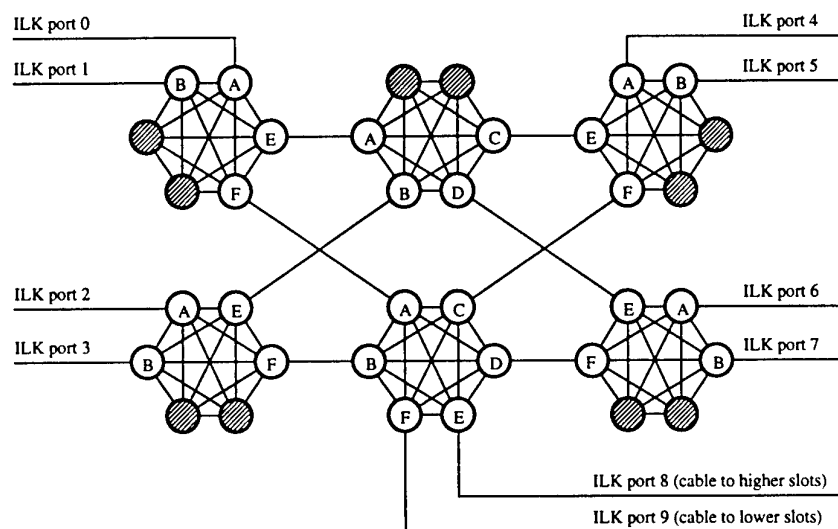
Figure 4.1: Configuration of the ILK8.

The ILK8 contains 6 crossbar chips which make it possible to interconnect eight PCBs and two neighbouring ILKs. To establish a connection through the ILK8, each data transfer starts with a routing word. The routing word contains up to nine port select codes and a broadcast / single mode bit. Each crossbar chip uses one port select code and the broadcast / single mode bit to determine the route through the crossbar chip. The coding table of the crossbar chips used on the ILK8, is given in Table 4.1.

Table 4.1:   *Crossbar chip coding table.*

Mode	Port select code	Output port
Single	7	A
Single	6	B
Single	5	C
Single	4	D
Single	3	E
Single	2	F
Single	1	E or (if busy) F
Single	0	F or (if busy) E
Broadcast	7	B, C, D if entered at A else A
Broadcast	6	A, C, D if entered at B else B
Broadcast	5	A, B, D if entered at C else C
Broadcast	4	A, B, C if entered at D else D
Broadcast	3	E
Broadcast	2	F
Broadcast	1	A, B, C, D, E
Broadcast	0	A, B, C, D, F

4.1.2   Clock

The clock jumpers on the ILKs must be set properly. Only one ILK generates the clock and all other ILKs and the PCBs connected to the ILKs receive the clock (see [MCIO96], page 234). If multiple ILKs are daisy chained, the ILK in the lowest VME slots generates the clock and the other ILKs receive the clock.

4.2   Transfer Data

4.2.1   Block / random access

A RACEway transfer consists of a routing word, address word, and one or more data elements as illustrated in Table 4.2.

Table 4.2:   *Data sequence.*

Route word[31:0]
Address word[31:0]
Data word 0[63:32]
Data word 0[31:0]
Data word 1[63:32]
....
Data word N[63:32]
Data word N[31:0]
Route word[31:0]
Address word[31:0]
Data word 0[63:32]
....

Two types of transfers are supported; block transfer and random access. Block transfers require one routing word and one address word to transfer multiple data elements. Random access transfers require a routing and address word for each data element transferred. Electrically, both types are the same; a random access transfer is simply a block transfer of size one. Functionally, however, the difference affects the overall RACEway throughput substantially. For block transfers, the following conditions must hold:

- the addresses of subsequent data words are sequential,
- each data element consists of 8 bytes,
- the block size does not exceed 2Kbyte (256K word),
- the master is not pre-empted by a higher priority transfer.

If any of these conditions are violated, the master must start a new transfer by driving a new routing and address word.

4.2.2   Transfer latency and throughput

The RACEway has a high throughput due to its pipelined structure. Since each crossbar is effectively a pipeline stage, once the route is established, data can be driven onto the RACEway in sequential clock cycles by the master (in case of write) or by the slave (in case of read). Table 4.3 and Table 4.4 illustrate optimal transfers with two crossbars separating the master and the slave.

Table 4.3:   Transfer latency for a write burst.

Cycle	Master	Crossbar 1	Crossbar 2	Slave
1	Routing word			
2				
3				
4			Routing word	
5				
6				
7				Routing word
8			Ready	
9		Ready		
10	Ready			
11	Data[63:32]	Address word		
12	Data[31:0]	Data[63:32]	Address word	
13	Data[63:32]	Data[31:0]	Data[63:32]	Address word
14	Data[31:0]	Data[63:32]	Data[31:0]	Data[63:32]
15	Data[63:32]	Data[31:0]	Data[63:32]	Data[31:0]
16	Data[31:0]	Data[63:32]	Data[31:0]	Data[63:32]
...				

Table 4.4:   *Transfer latency for a read burst.*

Cycle	Master	Crossbar 1	Crossbar 2	Slave
1		Routing word		
2				
3				
4			Routing word	
5				
6				
7				Routing word
8			Ready	
9		Ready		
10	Ready			
11		Address word		
12			Address word	
13				Address word
14				
15				
17				
18				
19			Data[63:32]	
20		Data[63:32]	Data[31:0]	
21	Data[63:32]	Data[31:0]	Data[63:32]	
22	Data[31:0]	Data[63:32]	Data[31:0]	
23	Data[63:32]	Data[31:0]	Data[63:32]	
...				

For read cycles, the figure assumes that the slave requires four clock cycles after receipt of an address to provide the first element of data. In general, routing of a path will take $3 \times C$ cycles of latency through C crossbars. The acknowledgement back will take C cycles. It will also take a few cycles for the master and the slave to respond to the signals. Therefore, the setup of the connection through the crossbar network will typically take $4 \times C + 4$ cycles for a route through C crossbars. For write operations, the data will directly follow after the route is established and 4 data bytes can be transferred every clock cycle. When the master wants to read from the slave, it will take an additional C cycles for the data before it reaches the master. Thus for read operations, it will take $5 \times C + 4$ cycles from starting a route to the first data reaching the master. In Table 4.5, some data rates are given for different data lengths.

Table 4.5:   *Effective data rate for various block lengths.*

Burst length (words)	Burst rate (Mbyte/sec)
1	4
2	9
4	47
16	100
64	139
256	154

The following three guidelines should be considered to help reduce transfer latencies:
- transfer blocks of data instead of random access,
- write rather than read,
- limit crossbar hops.

4.3   The RIC-RINO Chipset

4.3.1   Motivation
In the previous section, the following requirements are defined for the interface:
1. minimal communication overhead for the hardware subsystem,
2. minimal PCB area needed by the interface components,
3. where possible, make use of off-the-shelf components,
4. agree to the complete RACEway interlink specifications,
5. bidirectional data traffic,
6. applicable for various subsystems.

With the choice for the RIC-RINO chipset, compliance to most of these constraints is attained. At point 4, compliance to the complete RACEway interlink specifications, made a concession necessary. The RIC-RINO chipset is not able to act as a slave on the RACEway but only as a master. In return, the RIC-RINO chipset provides an easy to control interface. The drivers needed to control the interface will not be implemented on the RIC-RINO chipset but on an other RACEway node. Choosing for the RIC-RINO chipset gives the interface the following benefits:
- full 160Mbyte/sec bandwidth,
- master capabilities (used for data transport),
- minimal slave capabilities (only for initialisation and test purposes),
- compliance to the RACEway interlink protocol,
- existing drivers can be used which are located on an other node on the RACEway,
- data transport between the subsystem and the RACEway in both directions,
- universal interface to the subsystem.

4.3.2   Controlling the RIC-RINO chipset

Each RIC-RINO chipset must be controlled by a managing process. This managing process will run on an other node on the RACEway. All actions between the managing process and the RIC-RINO take place using the RACEway and must agree to the RACEway interlink protocol. The following actions take place to control the RIC-RINO chipset.

1. *Priming the RIC-RINO chipset*
    On power up, the node equipped with the RIC-RINO chipset starts up in an idle state and stays in this state until an other node on the RACEway primes him.

    A managing compute environment (CE) creates a linked list of command packets for the node with the RIC-RINO chipset.

    The managing CE primes the node with the RIC-RINO chipset; it writes the route and address of the first command packet in the linked list to the RIC-RINO chipset.

2. *Fetch a command packet*
    The node with the RIC-RINO chipset starts a read transaction to the received route / address. At this address, the RIC-RINO chipset will find its first command packet which contains the following info:

    *   word count (number of 8-byte words to transfer),
    *   control word,
    *   route and address of the data transaction,
    *   route and address of the next command packet.

3. *Start data transaction*
    The node with the RIC-RINO chipset starts a data transaction (as a master) on the RACEway and reads or writes to an other node until 'word count' words are transferred.

    When all words are transferred, the RIC-RINO chipset either goes to an idle state (like described in point 1) or fetches a new command packet on the next command route / address (like described in point 2). The decision whether it fetches a new command packet or not, depends on the go bit in the next command address.

4.3.3   Registers on the RIC-RINO chipset

The RIC-RINO chipset has several registers which can by written to or read from via the RACEway. These registers contain the word count, control word, data route, data address, command route and command address (as mentioned in the previous subsection) and a status word. The exact implementation of these

registers can be found in [MCIO96] page 344. The registers can only be approached by other nodes on the RACEway, not by the subsystem.

4.3.4   Data traffic from the subsystem to the interface

The data from the outside world (a subsystem) to the interface is buffered into a FIFO before it is transported to the RACEway as a burst operation. The signals mentioned in Table 4.6 control the data transport from the subsystem to the interface.

Table 4.6:   Signals used to transfer data to the RACEway.

Signal	Source	Function
ext_clock	subsystem	Clock from the subsystem. On the raising edge data is clocked into the receive FIFO. This clock can be asynchronous to the RACEway clock but is the same clock as used for transporting data from the interface to the subsystem.
/rx_valid	subsystem	Data is clocked into the receive FIFO only when /rx_valid is active.
/rx_sync	subsystem	Used to synchronise data frames and indicates the start of a data frame.
rx_suspend	XC3190	Asserted high when the FIFO is 127 words from full.
/rx_suspend	XC3190	Asserted low when the FIFO is 127 words from full.
rx_rdy	XC3190	The interface is ready to receive data when active.
/ovflow	XC3190	Indicates a receive FIFO overflow has occurred
pio[2:1]	XC3190 or subsystem	Programmable data bits used for software handshaking. The direction can be to or from the subsystem.
pioen[2:1]	XC3190	Indicates (when low) that the pio[2:1] bits are enabled
rx_data[31:0]	subsystem	Data from the subsystem. The 64 bits words are split over two clock cycles, first bits 63 to 32 then bits 31 to 0.

Reset of the receive FIFO

The receive FIFO can be reset either by the assertion of the /xresetio pin (which is coming directly from the RACEway) or by writing to 'receive FIFO reset' in the control register.

Writing data into the receive FIFO by the subsystem

Data of the subsystem is written into the receive FIFO under the following conditions:

- rx_rdy is active, indicating that the interface is ready to receive data,
- /rx_valid is active. The subsystem makes this signal active when it wants to clock data into the FIFO,
- /rx_sync conditions are met. If the 'sync wait' bit in the control register is not set, then /rx_sync is disabled and only the previous two steps are necessary to write data into the FIFO. Otherwise a single pulse on the /rx_sync signal is needed before data can be written into the FIFO. In this way, data is synchronised to the /rx_sync signal.

Start of data transfer to the RACEway
The RIC-RINO chipset counts the valid words that are clocked into the receive
FIFO. When the lesser of the value programmed into the word count register or 2k
words are clocked into the FIFO, data is read from the FIFO and written to the
RACEway as a burst operation. When the word count reaches 0, the next
command packet is fetched and operation continues.

Pio[2:1]
Two user programmable I/O bits are available for data tagging or other application
specific purposes. These bits may be individually programmed via the control
register to be either inputs or outputs. The values for these bits can be assigned
under linked list control to implement the tagging of command packets as they are
executed. For example, headers and data can be assigned different tags. The
pioen[2:1] bits indicate whether the pio[2:1] bits are enabled.

4.3.5   Data traffic from the interface to the subsystem
For data transport from the RACEway to the subsystem, the interface provides a
FIFO to buffer the data. In Table 4.7, the signals are provided to control the data
from the interface to the subsystem.

Table 4.7:   Signals used to transfer data to the subsystem.

Signal	Source	Function
ext_clock	subsystem	Clock from the subsystem. On the raising edge data is clocked from the FIFO to the subsystem. This clock can be asynchronous to the RACEway clock but is the same clock as used for data transport from the subsystem to the RACEway interface.
/tx_valid	QL12X16B	Valid signal used for the subsystem indicating that valid data has been read out of the transmit FIFO.
/tx_sync	QL12X16B	Used for the subsystem indicating the beginning of a new data frame.
/tx_suspend	subsystem	When /tx_suspend becomes active, reading out of the transmit FIFO will stop within four clock cycles of ext_clock.
tx_data[31:0]	FIFO	Data from the transmit FIFO. The 64 bits words are split over two clock cycles, first bits 63 to 32 then bits 31 to 0.
tx_rdy	subsystem	When active indicates that data can be read out of the transmit FIFO on the next edge of ext_clock.

Reset the transmit FIFO
The transmit FIFO can be reset either by the assertion of the /xresetio pin (which is
coming directly from the RACEway) or by writing to 'transmit FIFO reset' in the
control register.

Filling the FIFO by the RACEway
When the programmable almost full pin on the FIFO indicates that there are
enough available entries in the FIFO, a burst operation transfers data from the

RACEway to the FIFO to fill it up. The programmable almost full pin on the FIFO will become active when there are at least 576 empty locations in the FIFO. The size of the burst is the lesser of 2 kilobytes or the value programmed into the word count register.

Reading the FIFO by the subsystem
Data will be read out of the transmit FIFO when the following conditions are met:
- /tx_suspend is not active, the subsystem indicates that it is able to receive data,
- tx_rdy is active, indicating that the interface is ready to generate data,
- /tfefl and /tfefh are not active, indicating that the transmit FIFOs contain data.

Only when these conditions are met, a valid read occurs. The /tx_valid signal will go active in response to a valid data reads (/tx_valid won't go active for a sync marker).
It is not possible for the interface to immediately stop the reading out of the transmit FIFO when /tx_suspend becomes active. Up to two valid data reads can occur after /tx_suspend becomes active.

Sync mechanism
There are two sync mechanisms used by the RIC-RINO chipset, one is used to synchronise receive data and one is used to synchronise transmit data. The sync mechanism for the transmitted data works as follows. A controlling CE on the RACEway will activate the mechanism by writing the 'sync out' bit in the control register. When a sync out write occurs, the control chip will activate the signal SET_SYNC for one transmit FIFO write cycle. SET_SYNC is written into the transmit FIFO on a place which is not used by transmit data (for our case bit 17). The SET_SYNC signal will propagate through the transmit FIFO while reading the FIFO. After a surtain amount of read cycles, the SET_SYNC signal will arrive at the read side of the FIFO. At the read side, the signal is called INV_SYNC and will be supplied to the control chip. As INV_SYNC arrives at the control chip, it will activate /tx_sync to tell the subsystem that a new data frame will appear. Another thing that happens when INV_SYNC arrives at the control chip, is that the signal /tx_valid will be de-activated, indicating that the data presented on the transmit FIFO is not valid.

On transmit FIFO reset the programmable almost full flag is loaded with 0x240, which corresponds with 2304 bytes. This byte size is determined by allocating 2 kilobytes (512 32-bit entries) for data and 256 bytes (64 entries) for sync markers. This places an upper limit of 64 sync markers for every 256 data words which must be adhered to.

5.   FCM Controller Design

5.1   Data Transport from the RACEway to the FCM

Data has to be passed from the RACEway interface to the FCM by the FCM
controller. To manage this process, the FCM controller generates the appropriate
control signals.

5.1.1   Data transport from the RACEway interface to the FCM controller

The RACEway interface provides data and control signals to the FCM controller.
These signals are listed in Table 5.1.

Table 5.1:   *Signals intended for data transport from the RACEway interface to the FCM
controller.*

Signal	Direction	Function
TX_DIR	to FCM cnt	Indicates the direction of data transport through the RACEway interface. (not used in the COTS interface system).
OUT_MODE	to FCM cnt	Reserved by the RACEway interface for future use.
N_SYNC	to FCM cnt	Is active (low) when the first word of a new sequence can be read from the RACEway interface.
N_TX_VALID	to FCM cnt	Indicates (when low) that valid data has been read from the RACEway interface.
N_TX_SUSP	to Rw int.	When asserted (low) the RACEway interface will suspend generating data (always deserted in the COTS interface system).
TX_RDY	to Rw int.	Indication to the RACEway interface that the FCM controller is ready to accept data (always asserted in the COTS interface system).
PIO((1:0)	to FCM cnt	Programmable data bits for software handshaking (not used in the COTS interface system).
PIOEN(1:0)	to FCM cnt	Indicates (when low) that the PIO(1:0) bits are enabled (not used in the COTS interface system).
TX_DAT(31:0)	to FCM cnt	32 data bits.

The RACEway interface makes the N_SYNC signal active to indicate that the first
sample of a new sequence is present on its output. The FCM controller uses this
signal to initialise some processes. Directly after the sync signal the FCM interface
will start generating data, because N_TX_SUSP is always de-activated and
TX_RDY is always activated. Data will be become available using the protocol as
listed in Table 5.2. The data sequence starts with the length of the reference
sequence and the length of the data sequence. The FCM controller uses this
information to generate the REF_VALID and DAT_VALID signals. The five
program words, the reference and the data are passed to the FCM.

Table 5.2:   *Data format of the RACEway interface.*

Clock cycle	TX Data(31:0)
1	Sync
2	Reference sequence length
3	Data sequence length
4	zero
5	Program word 1
6	zero
7	Program word 2
8	zero
9	Program word 3
10	zero
11	Program word 4
12	zero
13	Program word 5
14..33	zero
34	reference(1)
35	data(1)
36	reference(2)
37	data(2)
.....	

5.1.2   Data transport from the FCM controller to the FCM

The 'FCM controller' provides several signals to the FCM and receives several signals from the FCM in order to control data transport to the FCM. The general signals are listed in Table 5.3, signals used for data transport to the FCM are listed in Table 5.4 and the function of the programming words is listed in Table 5.5.

Table 5.3:   *General signals to the FCM.*

Signal	Direction	Function
N_RESET	to FCM	Power up reset. When active (low) the FCM returns to an idle state. The 'FCM controller' passes the SYSRESET signal from the RACEway bus to the N_RESET input of the FCM.
FIRST_FCM	to FCM	The FCM with an active FIRST_FCM signal will process the first data sequence after a power up reset. Only one FCM will be used in the RT-SAR system, this is why the FCM controller will activate this signal always.

Table 5.4:   Signals used to transport data from the FCM controller to the FCM.

Signal	Direction	Function
DATACLK	to FCM	The up-going edge of this clock is used to clock data into the FCM.
ENIN	to FCM	A pulse on this FCM input indicates that the next input sequence must be loaded. This signal has only a function in a multi-FCM environment, for the RT-SAR this signal is continuously activated.
READY_IN	from FCM	A pulse on this FCM output indicates that all data is loaded. The FCM controller doesn't use this signal.
PROG	to FCM	Active when programming words are present on the data bus.
LDX	to FCM	A pulse on this FCM input indicates that valid data will start on the next clock cycle.
DAT_VALID	to FCM	The FCM controller makes this signal active when valid data is present.
STOP_X	to FCM	A pulse is generated on this input during the last valid sample in a sequence. This signal is not used by the FCM, the DAT_VALID signal will be used instead.
LDH	to FCM	A pulse on this FCM input indicates that valid reference data will start on the next clock cycle.
REF_VALID	to FCM	The FCM controller makes this signal active when valid reference data is present.
STOP_H	to FCM	A pulse is generated on this input during the last valid reference sample in a sequence. This signal is not used by the FCM, the REF_VALID signal will be used instead.
X_R(15:0)	to FCM	Real data.
X_I(15:0)	to FCM	Imaginary data.
EXP_IN(7:0)	to FCM	Exponent.
H_R(15:0)	to FCM	Real reference data.
H_I(15:0)	to FCM	Imaginary reference data and programming information.

Table 5.5:   Programming words.

Word	Bit	Function
1	15	Flag for generating preamble samples.
	14:0	Data length.
2	14:0	Reference length.
3	9	2 or 3 shifts for the FFT.
	8	Flag for processing the reference.
	7:0	Number of sequences.
4	14:0	Length of the output sequence.
5	7:0	Required exponent.

5.2 Data transport from the FCM to the RACEway

As soon as data comes available from the FCM it will be transported to the
RACEway using the FCM controller and the RACEway interface.

5.2.1 Data transport from the FCM to the FCM controller

The FCM generates data and control signals as listed in Table 5.6. These signals
are used by the FCM controller to load the data.

Table 5.6: Signals for data transport from the FCM to the FCM controller.

Signal	Direction	Function
KLOK_UIT	to FCM	Output data is generated on the up going edge of this clock signal.
ENOUT	to FCM	A pulse on this FCM input starts the data transmission from the FCM to the FCM interface. In the RT-SAR application the output data is wanted directly as it comes available, this is why the ENOUT input is continuously activated.
RDY_OUT	from FCM	Active during the read cycle of the last output sample of a sequence (not used in the RT-SAR application).
R_R(15:0)	from FCM	Real output data.
R_I(15:0)	from FCM	Imaginary output data.
EXP_R(7:0)	from FCM	Exponent for the output data.
N_RVALID	from FCM	Active (low) when valid data samples are presented at the output bus.

5.2.2 Data transport from the FCM controller to the RACEway interface

The FCM controller passes the FCM data to the RACEway interface using the
signals as listed in Table 5.7.

TNO report

TNO report

6.   Concluding Remarks

This report described the specifications of the result of the project "COTS Interface FCM".

The objective of the project was the specification, the design and the realisation of an interface between the high-speed real-time SAR system data bus (the RACEway) at the one side and the Fast Convolution Module (FCM) at the other side.

The design of the COTS (commercial-off-the-shelf) interface is generic. In the future, it can be used for other dedicated hardware subsystems as part of the multi-purpose real-time SAR processor environment.

The result of the project is successfully applied in the integration of the FCM in the multi-purpose real-time SAR processor. The FCM performs one of the critical SAR processing steps, the range compression. Using the FCM instead of commercially available DSP boards, saves at least a factor three in volume, power consumption and costs of the range compression hardware.

The COTS interface allows the integration of COTS products for flexible processing parts with dedicated hardware for "bulk-processing" parts. With this project TNO-FEL has shown that this "best-of-both-worlds" approach leads to a substantial reduction in volume, power consumption and costs of complex application specific signal processing systems.

7. References

[BIE97]   L.H.J. Bierens, "Hardware Design for Modern Radar Processing",
             IEE Electronics and Communications Engineering Journal,
             9(6):257-270, December 1997
[MCDG95]   Mercury Computer Systems, Developers guide, DG-00-14, 1995
[MCIO96]   Mercury Computer Systems, I/O Products, IP-00-10, 1996
[RASSP95]   Proceedings of the 2nd "Annual RASSP conference", July 1995
[VME82]   VMEbus manufacturers group, VMEbus specification manual,
             revision B, August 1982
[VITA94]   ANSI/VITA, American National Standard for RACEway interlink,
             VITA 5-1994

8. Signature

J.P. van Bezouwen
Group leader

L. Bierens
Project leader/Author

Appendix A   VME Bus Interface Requirements

This appendix describes the requirements of the VME bus interface for the interface between the FCM to RT-SAR. The goal is to specify an universally useable interface component, preferrably using COTS components.

GLOBAL VME Interface Requirements

The VMEbus interface will be used to handle relatively low-bandwidth information, mostly control and status information. The bus may also be used for data transfer, where RACEway compatible devices or boards do not exist, e.g. video display and SCSI interfaces. The VMEbus interface described here will be a generic control/status interface. We define the following requirements:
- conforming to VMEbus specifications,
- a number of 32-bits registers (read/write by both VMEbus and the subsystem),
- the VMEbus address space should be configureable by jumpers, which determine the address range used,
- compact design, preferably in a programmable device, such as an EPLD, or an off-the-shelf product.

The interface is basically a 'bridge' between the VMEbus on one side, and a proprietary RT-SAR-bus on the other. In the following sections, both will be detailed.

VMEbus address space
The VMEbus is described in detail in [VME82]. For the RT-SAR, the following applies:
- All system modules will be a slave device, with the exception of the main system controller, which will act as the master.
- The address bus of the VMEbus has 32 address lines, addressing 4096Mbytes of storage space. In this design, we use 32 bits data (dwords), which results in every fourth address being valid (i.e. $00,$04,$08,$0C,$10 ... $FFFFFFFC). This results in a 1 G dwords of address space.
- For each VMEbus interface a single, consecutive area will be reserved. It is possible, and adviseable to select an area as large as possible, to simplify the address decode logic. There is no harm in the fact that registers may have multiple "ghost" images within the address range.

Apart from the address lines A[31:1], the VMEbus also has provisions for a six-bit address modifier, AM5..AM0. The VME bus master does not provide support for these lines, and they must be ignored by the interface.

The Register Specification

The interface provides a number of registers, all 32-bits wide. The number of registers, and their use, will be determined by the subsystem designer. The interface provides a number of consecutive 32-bits read/write registers, beginning at a definable VME address.

The interface will be a VMEbus slave, the timing requirements of the VMEbus section of the interface are given in [VME82], pages 2-41 to 2-48. The only type of access is a standard read or write. DMA, interrupts and other special types are not supported.

The system controller
The system controller is the main CPU, running Solaris. It will need to be configured for the VMEbus interface to work correctly. This is done in the /kernel/drv/vme.conf file. The following entries need to be set:
vmewin=0x30000000
vmewin1=0x80000000
vme16d16=0x0,0x10000
vme16d32=0x0,0x10000
vme24d16=0x0,0x100000
vme24d32=0x0,0x100000
vme32d32=0x0,0x10000000
slavewin=0x00000000,0x100000,1;

The last entry, slavewin, defines a 256Kbyte window at address 0x80000000, which is to be used for one VMEbus interface. Each VMEbus interface should have its own "slavewin" entry. The vme.conf above also enables the VMEbus for all 16, 24 and 32-bits addressing modes. Only the 32bits (vme32d32) mode will be used by the interfaces.

RT-SAR Bus Interface

The other side of the interface is named the RT-SAR bus interface. It is designed specifically for the RT-SAR. The interface consists of a number of registers, at least one. There are a number of options, to be determined at the subsystem design. These options include:
- the base address (start address) of the interface,
- the length of the block the interface occupies on the VME bus,
- the number of 32-bits registers to implement,
- the function of the individual bits and registers (not needed for the VME interface, but for the subsystem itself).

Example Subsystem

In this section, we will describe an example subsystem. It has three control
registers and one status register. The fictive subsystem is a temperature monitor,
which measures a temperature, and can signal too hot and too cold errors. It is
assigned the VME address area starting at $80000000, with a length of 256k byte.
The ending address will therefor be $8003FFFF. Its register layout is defined in
Table A.1.

Table A.1:   *Register layout of the subsystem example.*

Address	name	function
$80000000	CONTROL	Determines the mode of the subsystem
$80000004 ·	STATUS	Shows the measured value
$80000008	LIMIT_HI	Sets the upper limit.
$8000000C	LIMIT_LO	Sets the lower limit
$80000010..8003FFFC		Reserved, contains "ghosts" of the other registers

The individual functions and uses of the registers are not important for the
interface design. What is important, is the range of addresses to decode. From the
table above, we can extract that all addresses can be defined as

%1000.0000.0000.00xx.xxxx.xxxx.xxxx.NN00.

This shows, that only address lines A31 through A18 are to be considered for a
"interface select", A17..A4 are don't care, and should not be incorporated in the
address decoder. A3 and A2 determine which register is to be accessed, and A1
and A0 are always 0, as goes for all 32-bits accesses. From this, the complete
address decoder can be designed.

References

[VME82]   VMEbus manufacturers group, VMEbus specification manual,
             revision B, August 1982

ANNEX   VME Bus Connector

The VMEbus uses two 96-pin 603-2-IEC-CO96-M connectors. The upper one is
designated P1, the lower P2. Table A.2 and Table A.3 show the connector pinouts
for P1 and P2, respectively.

Table A.2: P1 connector pinout.

Col.	Row A signal name	Row A signal type	Row B signal name	Row B signal type	Row C signal name	Row C signal name
01	D00	3S	BBSY*	OC	D08	3S
02	D01	3S	BCLR*	TP	D09	3S
03	D02	3S	ACFAIL*	OC	D10	3S
04	D03	3S	BG0IN*	TP	D11	3S
05	D04	3S	BG0OUT*	TP	D12	3S
06	D05	3S	BG1IN*	TP	D13	3S
07	D06	3S	BG1OUT*	TP	D14	3S
08	D07	3S	BG2IN*	TP	D15	3S
09	GND	POW	BG2OUT*	TP	GND	POW
10	SYSCLK	TP	BG3IN*	TP	SYSFAIL*	OC
11	GND	POW	BG3OUT*	TP	BERR*	OC
12	DS1*	3S	BR0*	OC	SYSRESET*	OC
13	DS0*	3S	BR1*	OC	LWORD*	3S
14	WRITE*	3S	BR2*	OC	AM5	3S
15	GND	POW	BR3*	OC	A23	3S
16	DTACK*	OC	AM0	3S	A22	3S
17	GND	POW	AM1	3S	A21	3S
18	AS*	3S	AM2	3S	A20	3S
19	GND	POW	AM3	3S	A19	3S
20	IACK*	OC	GND	POW	A18	3S
21	IACKIN*	TP	SERCLK	RES	A17	3S
22	IACKOUT*	TP	SERDAT	RES	A16	3S
23	AM4	3S	GND	POW	A15	3S
24	A07	3S	IRQ7*	OC	A14	3S
25	A06	3S	IRQ6*	OC	A13	3S
26	A05	3S	IRQ5*	OC	A12	3S
27	A04	3S	IRQ4*	OC	A11	3S
28	A03	3S	IRQ3*	OC	A10	3S
29	A02	3S	IRQ2*	OC	A09	3S
30	A01	3S	IRQ1*	OC	A08	3S
31	-12V	POW	+5V STBY	POW	+12V	POW
32	+5V	POW	+5V	POW	+5V	POW

Table A.3: *P2 connector pinout.*

Col.	Row A signal name	Row A signal type	Row B signal name	Row B signal type	Row C signal name	Row C signal name
01	user		+5V	POW	user	
02	user		GND	POW	user	
03	user		RESERVED	RES	user	
04	user		A24	3S	user	
05	user		A25	3S	user	
06	user		A26	3S	user	
07	user		A27	3S	user	
08	user		A28	3S	user	
09	user		A29	3S	user	
10	user		A30	3S	user	
11	user		A31	3S	user	
12	user		GND	POW	user	
13	user		+5V	POW	user	
14	user		D15	3S	user	
15	user		D16	3S	user	
16	user		D17	3S	user	
17	user		D18	3S	user	
18	user		D19	3S	user	
19	user		D20	3S	user	
20	user		D21	3S	user	
21	user		D22	3S	user	
22	user		D23	3S	user	
23	user		D24	3S	user	
24	user		D25	3S	user	
25	user		D26	3S	user	
26	user		D27	3S	user	
27	user		D28	3S	user	
28	user		D29	3S	user	
29	user		D30	3S	user	
30	user		D31	3S	user	
31	user		GND	POW	user	
32	user		+5V	POW	user	

TNO report

The signal types are explained in Table A.4. Table A.5 describes the signals. A *
postfix denotes a signal that is active low, i.e. the condition is true when the signal
level is below 0.8V, and false when the level is above 2.4V.

Table A.4:   Signal types.

Signal type	Description
OC	Open Collector TTL
3S	Three state TTL
TP	Totem pole TTL
POW	Power/ground signal
RES	Reserved

Table A.5:   *Signal descriptions.*

Signal name	Description
ACFAIL*	The AC input to the system power supply is no longer being supplied.
IACKIN*	Indicates to the board that an interrupt acknowledge cycle is in progress. This, and IACKOUT are to be daisy-chaned between boards, across the backplane.
IACKOUT*	(see IACKIN*)
AM[5:0]	Address Modifier - Gives additional information on the cycle-type. Is not used in the RT-SAR system.
AS*	Address strobe - indicates a valid address is on the bus
A[23:1]	Address bus, bits 1..23.
BBSY*	Bus Busy - the current DTB master is using the bus
BCLR*	Bus Clear - the bus arbitrator requests the current DTB to release the bus.
BERR*	Bus Error - A bus slave indicates an unrecoverable bus error, and the current cycle must be aborted.
BG[3:0]IN*	Bus Grant in - Bus grant in/out are daisy-chained between boards, across the backplane. Indicates that this board may become the next master
BG[3:0]OUT*	(see BGIN), indicates that the next board may become bus master
BR[3:0]*	Bus Request - Indicates that a DTB master requires access to the bus
DS0*	Indicates data transfer will occur on data lines D00..D07.
DS1*	Indicates data transfer will occur on data lines D08..D15
DTACK*	Data Transfer Acknowledge - the falling edge, generated by the slave, indicates that valid data is available during a read cycle, or data is accepted during a write cycle.
D[31:0]	Data bus, 32 bits wide.
IACK*	Interrupt Acknowledge - generated by a bus master when it services an intterrupt request.
IRQ[7:1]*	Interrupt request - asserted by a slave when it signals an interrupt. Proritised.
SERCLK	Reserved, will be used for serial communications clock in future implementations
SERDAT	Reserved, will be used for serial communications data in future implementations
SYSCLK	System clock 16MHz, independent of any processor or subsystem clocks.
SYSFAIL*	Signal that may be generated by any module, signalling a fatal system failure
SYSRESET*	System reset, will reset all modules and processors
WRITE*	Indicates a write cycle is in progress
LWORD*	Indicates current cycle is long-word (32 bits) wide.
RESERVED	Reserved line - no connection is to be made.
user	User line - Used for RACEway bus in the RT-SAR system.

REPORT DOCUMENTATION PAGE
(MOD-NL)

1. DEFENCE REPORT NO (MOD-NL)	2. RECIPIENT'S ACCESSION NO	3. PERFORMING ORGANIZATION REPORT NO
TD98-0138		FEL-98-A052

4. PROJECT/TASK/WORK UNIT NO	5. CONTRACT NO	6. REPORT DATE
27204	A97KLu735	July 1998

7. NUMBER OF PAGES	8. NUMBER OF REFERENCES	9. TYPE OF REPORT AND DATES COVERED
41 (incl 1 appendix, excl RDP & distribution list)	6	Final

10. TITLE AND SUBTITLE

COTS Interface FCM

11. AUTHOR(S)

L.H.J. Bierens, C. van 't Wout, P.A.M. Stijnman

12. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

TNO Physics and Electronics Laboratory, PO Box 96864, 2509 JG The Hague, The Netherlands
Oude Waalsdorperweg 63, The Hague, The Netherlands

13. SPONSORING AGENCY NAME(S) AND ADDRESS(ES)

DMKLu/MXS
Binckhorstlaan 135, 2516 BA The Hague, The Netherlands

14. SUPPLEMENTARY NOTES

The classification designation Ongerubriceerd is equivalent to Unclassified, Stg. Confidentieel is equivalent to Confidential and Stg. Geheim is equivalent to Secret.

15. ABSTRACT (MAXIMUM 200 WORDS (1044 BYTE))

This report describes the result of the project 'COTS Interface FCM'. The objective of the project was the specification, the design and the realisation of an interface between the high-speed real-time SAR system data bus (the RACEway) at one side and the Fast Convolution Module (FCM) at the other side. The design of the COTS (commercial-off-the-shelf) interface is generic. In the future, it can be used for other dedicated hardware subsystems as part of the multi-purpose real-time SAR processor environment. The result of the project is successfully applied in the integration of the FCM in the multi-purpose real-time SAR processor. The FCM performs one of the critical SAR processing steps, the range compression. Using the FCM instead of commercially available DSP boards, saves at least a factor three in volume, power consumption and costs of the range compression hardware. The COTS interface allows the integration of COTS products for flexible processing parts with dedicated hardware for 'bulk-processing' parts. With this project TNO-FEL has shown that this 'best-of-both-worlds' approach leads to a substantial reduction in volume, power consumption and costs of complex application specific signal processing systems.

16. DESCRIPTORS	IDENTIFIERS
Real-Time Computation Synthetic Aperture Radar Data Processing Signal Processing	Digital Signal Processing

17a. SECURITY CLASSIFICATION (OF REPORT)	17b. SECURITY CLASSIFICATION (OF PAGE)	17c. SECURITY CLASSIFICATION (OF ABSTRACT)
Ongerubriceerd	Ongerubriceerd	Ongerubriceerd

18. DISTRIBUTION AVAILABILITY STATEMENT	17d. SECURITY CLASSIFICATION (OF TITLES)
Unlimited distribution	Ongerubriceerd

Distributielijst

1.	DWOO
2.	HWO-KM*
3.	HWO-KL*
4.	HWO-KLu
5.	HWO-CO*
6.	DMKLu/MXS, t.a.v. Drs. G.J. de Wilde
7.	DM&P TNO-DO
8.	Directeur TNO-PML*
9.	Directeur TNO-TM*
10.	Accountcoördinator KLu*
11 t/m 13.	Bibliotheek KMA
14.	Directeur TNO-FEL
15.	Adjunct-directeur TNO-FEL, daarna reserve
16.	Archief TNO-FEL, in bruikleen aan MPC*
17.	Archief TNO-FEL, in bruikleen aan Accountmanager KLu*
18.	Archief TNO-FEL, in bruikleen aan Dr.ir. J.L.J. de Sonneville
19.	Archief TNO-FEL, in bruikleen aan Ir. J.P. van Bezouwen
20.	Archief TNO-FEL, in bruikleen aan Dr.ir. L.H.J. Bierens
21.	Archief TNO-FEL, in bruikleen aan Ing. C. van 't Wout
22.	Archief TNO-FEL, in bruikleen aan Ing. P.A.M. Stijnman
23.	Archief TNO-FEL, in bruikleen aan Ing. J.C.P. Bol
24.	Archief TNO-FEL, in bruikleen aan Ir. A.W.P. van Heijningen
25.	Archief TNO-FEL, in bruikleen aan Ir. M.J. de Bijl
26.	Archief TNO-FEL, in bruikleen aan Ing. W.H.M. Groen
27.	Documentatie TNO-FEL
28.	Reserve

Indien binnen de krijgsmacht extra exemplaren van dit rapport worden gewenst door personen of instanties die niet op de verzendlijst voorkomen, dan dienen deze aangevraagd te worden bij het betreffende Hoofd Wetenschappelijk Onderzoek of, indien het een K-opdracht betreft, bij de Directeur Wetenschappelijk Onderzoek en Ontwikkeling.

* Beperkt rapport (titelblad, managementuittreksel, RDP en distributielijst).