A Compact Robin Using the SHarc (CRUSH)

Draft version 0.97, September 18, 1998

P. P. M. Jansweijer, G. N. M. Kieft, J. C. Vermeulen
NIKHEF, Amsterdam

1. Introduction

Compactness of design and relatively low power dissipation are essential properties for ROBIns when it is desired to put many of them together on a single board. In the CRUSH design study these aspects are addressed, while at the same time it has been attempted to provide good programmability with respect to the required buffer management and also to provide a good testability of the design. How to build a ROB Complex with these ROBIns is also an element of this study.

In this document the design choices are discussed and the design is described. Familiarity with the functionality required is assumed. Measurement results will be added when available.

2. Basic assumptions

Event data is sent to the ROBIn at a maximum rate of 100 kHz through a 100 MByte/s data link that is compliant with the S-Link specification. The event data has a format similar to that described in the "ATLAS Read-Out Link Data Format". In general, event data is packed between a header and a trailer. The header is starting with a Begin Of Block (BOB) word and the trailer is ending with an End Of Block (EOB) word. Important event information (like the LVL1 event ID) is packed in the header as well. The data has to be stored in a buffer memory and to be provided when a request for it is received. The data has to be discarded from the buffer memory when a command for doing so is received.

3. Design considerations

The input data has to be scanned with a speed of 100 MByte/s at least for BOB and EOB words and has to be stored. Event id's have to be found in the data. The information extracted has to be stored for later usage when requests or discard commands for event fragments are received. When a discard command is received the memory occupied by the event fragment to be discarded should be freed. As discard commands can arrive a long time after the event data has been received, a strategy for handling memory fragmentation should be implemented.

A processor that is fast enough can read the event data directly from the S-link connector via an intermediate FIFO (for preventing loss of data), inspect the data and store it in a memory. However, the BOB and EOB words are signalled with a 33rd bit, which cannot be handled by 32-bit processors. As all data needs to be inspected the

CPU needs to be able to check the data with a speed of 100 MByte/s, with enough CPU power left for executing the remaining tasks. It is not excluded that a single processor solution (without special purpose hardware) is possible, but when we started our study this option did not look to be feasible.

Inspection of the incoming data with special purpose hardware (in particular with a FPGA) that also takes care of transporting the data from the S-link connector to a buffer memory leads to a reduction of the required processing power. In principle implementing all the functionality required in dedicated hardware is possible. A drawback of the latter approach is less flexibility compared to implementation of most of the required functionality in software.

Our conclusion is that a combination of an embedded processor and dedicated hardware offers a good solution. In view of the desired programmability and flexibility the patterns that are looked for in the event data can be written by the processor in the FPGA that is used. To limit the complexity of the FPGA the buffer memory is handled as a circular memory. The FPGA does not take care of management of the buffer memory, so that there is no need to maintain queues of free and used pages and / or to manage a linked list structure in the buffer memory. With a circular memory of course the danger exists that data is overwritten unnecessarily, as there could be free memory at another place in the buffer then at the address to which the buffer pointer is pointing. However, for a large enough buffer memory, it will be feasible to move this data from the buffer memory to an overflow buffer under processor or DMA control. In this case only for a small fraction of the events a decision on acceptance or rejection of the data arrives so late that the data is in danger of being overwritten. Furthermore the data now only needs to be transported and does not need to be inspected so a normal DMA controller could be used. By providing the event start addresses in the Buffer Memory together with selected words from the event data (in particular the LVL1 event ID) direct to the processor there is no need during normal operation to read from the Buffer Memory other than for transporting event data out of the memory. By organising this “summary information” in data blocks of a fixed length transport of this data under DMA control (ideally using auto-initialise mode) is possible.

The choice of the processor is an important issue. Obvious requirements are sufficient processor speed and support for inter-processor communication (via a suitable network, possibly indirectly via shared memory with a host processor with network interfacing). Also sufficient bandwidth for communication with the FPGA and the buffer memory, for communication of the CPU with its program and data memories and for inter-processor communication should be available. Furthermore, in view of our objectives compactness and low power dissipation also play an important role.

The SHARC DSP from Analog Devices has been chosen as processor for our design study. The SHARC has a large internal SRAM memory (512 kByte). This memory consists of 2 equally sized blocks, which each are implemented as dual-port memory, with one port connected to the I/O bus while the other port is shared by the two internal busses (PM and DM bus). Accesses via the two ports do not interfere with each other: every clock cycle (25 ns for the 40 Mhz SHARC used) an access is possible via each port. The internal memory is expected to be large enough for storing all code and data (including use as overflow buffer memory).

The SHARC also has six 4-bit parallel communication links, which each provides a maximum bandwidth of 40 MByte/s. These links allow for connecting to one or more other SHARCs. A ROBcomplex should be straightforward to construct from a number of SHARC based ROBins, connected via SHARC links to one or more SHARCs, in turn connected via PCI bus to a general-purpose CPU board with the required network interfaces. The high bandwidth of the links also allows to transport data from the overflow buffer memory in the internal memory of the SHARC to another overflow buffer in the memory of another SHARC or in the memory of the general-purpose CPU board.

The SHARC has a 48-bit parallel data bus, which can be used for interfacing to the FPGA and the Buffer Memory. In practice only 32-bits of the bus can be used, as only instructions and floating-point numbers can be transported as 48-bit quantities.

Due to the size of the internal memory, the communication links, the compactness of the SHARC itself and its low power dissipation, a compact design is possible. Due to the DMA facilities of the SHARC, its fast interrupt handling and due to the memory structure which allows DMA transfer into and out of the internal memory fully concurrent with CPU access, it is also expected that a design based on the SHARC can satisfy the requirements with respect to speed.

Important considerations for the choice of the FPGA are the speed at which it should operate and the functionality to be implemented. In particular the implementation of FIFO memory blocks then presents a problem. With respect to the speed synchronous operation with the 40 MHz SHARC is desirable in view of efficient communication between FPGA and SHARC. Synchronisation with the S-link clock is not possible, as this clock is not guaranteed to be always present. This dictates a 40 MHz clock speed for the FPGA, and for parts of it 80 Mhz in view of the Buffer Memory design. The CRUSH design makes use of the Altera FLEX 10k100A device, for which it was possible to produce a design that satisfies the requirements. The possibility to use other devices, e.g. from Xylinx, has been studied, but not in great detail in view of the time required for doing so. The conclusion was that the Altera technology is a good choice for our application and that there is not an obviously better alternative.

For the choice of the buffer memory technology again the desire to build a compact system is important. It is desirable to avoid using FIFO's, as may be required when using memory that is most efficiently accessed when blocks of data are written or read. High-speed memory is needed that should allow storage of the incoming event data (32-bits data arrive at 27 MHz at maximum for an S-link implementation based on FibreChannel hardware) and retrieval of a fraction of this data. With synchronous

SRAMs fast memory can be built (faster than with asynchronous SRAM), but a wait cycle needs to be inserted when switching between reading and writing. This can be avoided with ZBT (Zero Bus Turnaround) RAM. This type of memory is therefore used. A dual-port memory has been constructed (by running the memory on a double clock of 80 MHz) with one access per 25 ns possible on both ports, allowing for a 40 MHz input rate for 32-bits data, which results in a maximum input bandwidth of 160 MByte/s. As the buffer memory is running on an 80 MHz clock and due to the properties of a ZBT-RAM, all memory interfacing needs to be strongly pipelined. Therefore it takes the SHARC four system clock cycles to write or read data in or out of the Buffer Memory, hence the bandwidth for the SHARC to the Buffer Memory is 40 MByte/s.

4. Design of the CRUSH ROBIn

In Fig. 1 a block diagram of the design of the CRUSH ROBIn is presented. Four major parts can be distinguished:

- the S-link input and S-link FIFO,
- the FPGA, through which all the data flows from the S-link into the Buffer Memory and which provides summary and status information and signals various conditions to the SHARC,
- the Buffer Memory,
- the SHARC processor.

In this section various aspects of the design are described.

4.1. The S-link input and S-link FIFO

The design has a standard S-Link connection, via which the input event data is received. For the S-Link, the CRUSH is a Read-Out Motherboard (ROMB) and should be equipped with a Link Destination Card (LDC), a small piggyback card plugged onto the S-Link connector. Data from the S-link enters through the S-link connector and is stored in a FIFO. The write clock is taken from the S-Link clock domain (LCLK). At the other side of the FIFO, data is read out from the FIFO with a read clock taken from the clock domain of the FPGA and of the SHARC (CLK). In the present design the main task of the FIFO is to split those two clock domains. Unfortunately at present this FIFO can not reside in the FPGA (an Altera EPF10K100ARC240-1) because the FPGA does not support implementation of true Dual Ported Memory. However, in the near future there will be support in new FPGA's for the implementation of true Dual Ported Memory. This will make it possible to shift the S-Link FIFO into the FPGA, which further decreases power consumption and board space.

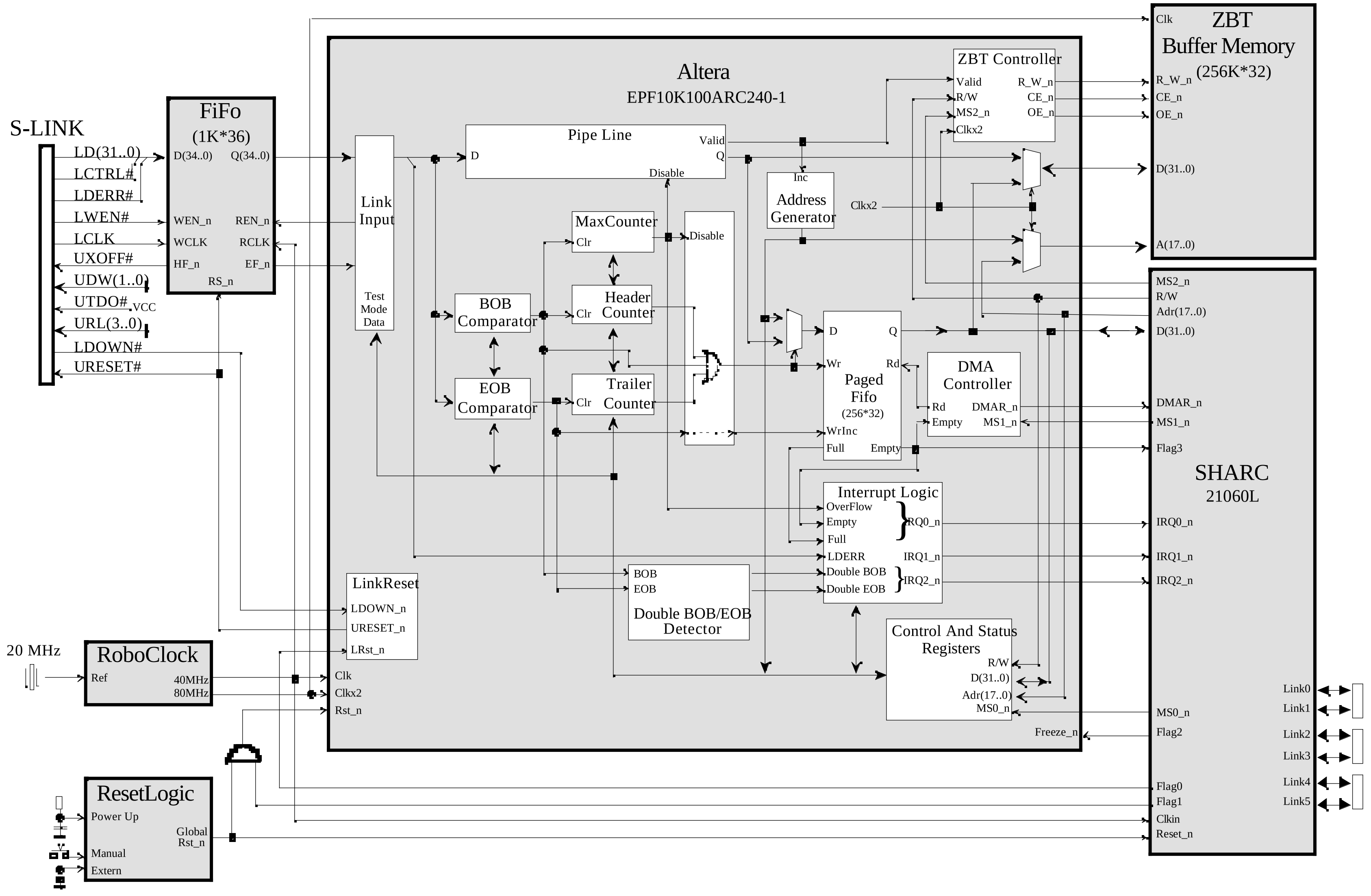

Fig. 1 Block diagram of the CRUSH

When the FIFO becomes Half Full, it signals the S-Link to stop data transmission by asserting UXOFF#. It may take the S-Link some time before actually stopping the data transmission. A second task of the FIFO therefore is to store S-Link data which arrives after asserting UXOFF#, for which 512 words are available in the FIFO. However, in the CRUSH design an UXOFF# condition should never occur because the FIFO is always read faster than data is written into it, except when not enough free space in the CRUSH Buffer Memory is available. In this case the SHARC has the possibility to stop the data flow into the FPGA (SHARC flag 2 driving the ‘Freeze_n’ signal), so that the SHARC can move data that is already in the Buffer Memory to an overflow buffer until there is enough free space. The input FIFO will then fill up and could signal the S-Link to stop transmitting data. This should not occur during normal operation when the Buffer Memory is large enough (see also section 3).

The S-Link is configured for 32-bit operation only (UDW(1..0) = “00”). Because the CRUSH itself has test facilities, the test data from the S-Link will not be sent by the LDC when the S-Link is in test mode (UTDO# = ‘1’). The User Return Lines of the S-Link are not used and tied to logical zero (URL(3..0) = “0000”). In future designs it should be possible to drive these lines under software control. As specified in the S-Link Interface Specification, a simple state machine in the FPGA controls resetting the S-Link.

4.2. The FPGA

The FPGA has the following functions:

- generation of addresses used for storing the data in the Buffer Memory,
- providing information obtained from the incoming data stream and on the addresses at which event fragments are stored in the Buffer Memory. Which information is to be extracted can be programmed. The extracted information is stored in a special FIFO inside the FPGA (the “paged FIFO”),
- signalling of error conditions that can occur, such as errors in the data stream or Buffer Memory overflow.

Data read out of the FIFO travels through a pipeline in the FPGA and is written in the Buffer Memory. The clocked pipeline provides delay, so that enough time is available to process the data in the FPGA.

4.2.1.Buffer Memory address generation

The Buffer Memory acts as a circular buffer for the data arriving via the S-link. An Address Generator in the FPGA supplies contiguous addresses to store the data in the Buffer Memory. When necessary the SHARC can free up the Buffer Memory by transferring data to its internal memory (see section 3). Storing data in the Buffer Memory stops once more words are received for an event fragment than a certain maximum stored in a programmable counter (MaxCounter in fig. 1). This will prevent that events in the Buffer Memory are overwritten when an extreme long (probably faulty) event fragment is received. When more words are received than the value stored in MaxCounter, an interrupt (IRQ0_n) can be generated for the SHARC to signal this error condition. The counter is 18-bits wide and can be programmed for any value. The interrupt can be enabled or disabled under software control.

When there is an error in the data stream that is signalled by a LDERR# on the S-Link, an interrupt signal (IRQ1_n) is sent to the SHARC. The SHARC can read the Buffer Memory address at which the erroneous data word is stored (see section 5).

The incoming data is fed to a BOB-comparator and an EOB-comparator. These comparators signal whenever there is a BOB or EOB data pattern in the data stream. Both comparators can be programmed with a pattern and a mask. This means that any data pattern (LD(31..0) and LCTRL#) can serve as a BOB or EOB data pattern. Programming is done by writing the patterns and masks desired to the Control and Status Registers (see section 5). Note that this should be done during initialisation of the CRUSH.

Whenever an event is not terminated with an EOB condition (double BOB) or whenever a BOB condition is missing (double EOB) an interrupt (IRQ2_n) can be generated on the SHARC. The Buffer Memory address of the double BOB or EOB condition can be read by the SHARC. By looking into the Buffer Memory, the SHARC can distinguish between a missing BOB or a missing EOB condition.

4.2.2. Extraction of information from the incoming data stream

The FPGA will copy relevant information from the data stream (in particular the LVL1 event ID), together with the address of the BOB in the Buffer Memory, to one of 32 pages in the Paged FIFO. .Each page in the Paged FIFO consists of eight 32-bit words and can be read out of the Paged FIFO by the SHARC under DMA control using hardware handshaking between the FPGA and the SHARC (with the DMAR input signalling availability of data and the MS1 output signalling a read cycle in which data from the FIFO is read). For readout the Paged FIFO behaves as a normal FIFO. The summary information for one event always is stored in a single page.

Whenever a BOB data pattern is found in the data stream, a 5-bit Header Counter is started. This counter increments on every data word in the data stream. The content of this counter is compared to three programmable 5-bit preset values. If the counter value and one of the preset values match, the corresponding data word in the header of the data stream is also written into the Paged FIFO. In the same way one or both of the data words directly in front of the EOB data pattern can be copied into the Paged FIFO. The possibility to select 5 words in stead of one or two may seem to be overly generous. However, the FPGA has a sufficient amount of resources available to make this possible without compromising the design.

Whenever there is a FIFO Full or Empty condition for the Paged FIFO, an interrupt (IRQ0_n) can be generated on the SHARC. Both conditions can be enabled or disabled under software control. Moreover, the Empty condition of the Paged FIFO can be polled by the SHARC by looking at the status of its flag 3.

4.2.3. Control and Status Registers and Interrupt Logic

A set of 21 registers controls the FPGA and provides status information to the SHARC. The registers reside in the MS0 address space of the SHARC. Under control of these registers the following conditions may be signalled to the SHARC by means of its interrupt lines:

- IRQ0: paged FIFO empty (this condition is also signalled to the SHARC via its flag 3), paged FIFO full, Event Overflow (more words after BOB pattern than the maximum allowed, specified in one of the registers).

- IRQ1: LDERR bit of the S-link set, one of the registers contains the buffer memory address at which the erroneous data word is stored, a bit signals whether more than one error occurred,

- IRQ2: two BOB words without an intermediate EOB word, or two EOB words without an intermediate BOB word were seen.

The bit patterns to be recognised as Begin Of Block word (BOB) and as End Of Block word (EOB) can be programmed as a 33-bit pattern and a 33-bit mask. The 33 bits are organised as two 32-bit words; in the last word, only bit 0 is significant. This single bit signals whether or not the first 32-bit word needs to be an S-Link control word or not.

Registers are available for specifying which of 3 out of 30 words that follow directly after the BOB data pattern will be copied to the Paged FIFO. Also the two last words before an EOB data pattern can be copied to the paged FIFO.

To test the hardware of the CRUSH, it is possible to “disconnect” the S-Link as a data source and put test data into the CRUSH, under software control of the SHARC. In Test Mode, the test data flows trough the FPGA to the Buffer Memory and to the SHARC as if it came from the S-Link. Results can be monitored by reading the Paged FIFO and by reading the Buffer Memory content.

4.3. The ZBT Buffer Memory

The Buffer Memory needs the functionality of a Dual Ported Memory. At one port, data coming from the S-Link is continuously written into the memory. At the other port, the SHARC must be able to read (and write for test purposes) the data out of the Buffer Memory for further processing.

The Zero Bus Turnaround (ZBT) Buffer Memory is running on a double clock (Clkx2), so that a Dual Ported Memory is emulated. The ZBT Controller in the FPGA controls this process of "time sharing". The bandwidth for the S-Link data to the Buffer Memory is 160 MByte/s. The bandwidth for the SHARC to the Buffer Memory is 40 MByte/s (due to the necessary pipelining in the FPGA).

The SHARC can read or write the Buffer Memory by using its memory region MS2. This makes it possible to test the memory and/or to put known event data in it.

4.4. The SHARC

The SHARC can access the Control and Status Registers in the FPGA in its memory region MS0. The SHARC can extract the event summary information from the Paged FIFO in the FPGA in memory region MS1. DMA with hardware handshaking can be used for transporting the data from the Paged Fifo (always 8 words). Using memory region MS2, it can read or write the Buffer Memory with a bandwidth of up to 40 MByte/s. Communication of the SHARC to the outer world (level-2 trigger) is done over 6 link ports. Those ports can transfer data with a bandwidth op 40 MByte/s each. The link ports can operate under DMA control.

4.5. The Reset Logic

The CRUSH has several reset possibilities. First of all, there is a global reset ('Global Rst_n'). This signal resets the SHARC and the FPGA (including the Pipeline, Address Generator and the Paged FIFO). The global reset becomes active during power-up, a manual reset or an external reset. Note that the global reset does not reset the S-Link and the S-Link FIFO. Resetting the S-Link and the S-Link FIFO can be done once the CRUSH is up and running. The SHARC can then reset the S-Link and the S-Link FIFO using flag 0. The SHARC can also reset the FPGA (including the Pipeline, Address Generator and the Paged FIFO) using flag1.

4.6. The RoboClock

The CRUSH needs a 40 MHz 'system clock' (Clk) and an 80 MHz double clock (Clkx2). Those clocks are generated using a RoboClock and a quartz crystal.

5. Control and Status Registers in the MS0 address space of the SHARC

Table 1 contains a specification of the Control and Status registers.

5.1 BOB Pattern and Mask Registers (00H to 03H)

Register 00H contains the 32-bit pattern and register 01H the control bit that must match the BOB data on the S-Link. A mask (register 02H and 03H) defines the bits that must match. If a mask bit is set to ‘1’ the corresponding bit in the pattern register must match the BOB data on the S-Link. If they do, a BOB is signalled. If a mask bit is set to ‘0’ the corresponding bit in the pattern register may be different from the bit in the S-Link data. Reading back registers 00H to 02H yields the value written into them.

Address	Function	Write	Read
00H	BOB Pattern	nnnnnnnnH	nnnnnnnnH
01H	BOB Control Bit Pattern	xxxxxxxpH	0000000pH
02H	BOB Mask	nnnnnnnnH	nnnnnnnnH
03H	BOB Control Bit Mask	xxxxxxxpH	0000000pH
04H	EOB Pattern	nnnnnnnnH	nnnnnnnnH
05H	EOB Control Bit Pattern	xxxxxxxpH	0000000pH
06H	EOB Mask	nnnnnnnnH	nnnnnnnnH
07H	EOB Control Bit Mask	xxxxxxxpH	0000000pH
08H	Put BOB Address	xxxxxxqnH	000000qnH
09H	BOB Word Flag 0	xxxxxxqnH	000000qnH
0AH	BOB Word Flag 1	xxxxxxqnH	000000qnH
0BH	BOB Word Flag 2	xxxxxxqnH	000000qnH
0CH	EOB Word Flag 0	xxxxxxxpH	0000000pH
0DH	EOB Word Flag 1	xxxxxxxpH	0000000pH
0EH	Maximum Event Size	xxxqnnnnH	000qnnnnH
0FH	Interrupt Control IRQ0	xxxxxxqnH	000000qnH
10H	LDERR_n Interrupt IRQ1	xxxxxxxxH	nnnnnnnnH
11H	BOB / EOB Double Interrupt IRQ2	xxxxxxxxH	nnnnnnnnH
12H	Test Control Register	xxxxxxxqH	0000000qH
13H	Test Link Data Register	nnnnnnnnH	nnnnnnnnH
14H	Test Link Control Register	xxxxxxxqH	0000000qH

Remarks:
‘n’ is any hexadecimal number
‘x’ is don’t care
‘p’ is either 00H or 01H
‘q’ is 00H, 01H, 02H or 03H

Table 1 Control and Status Registers in the MS0 address space of the SHARC

5.2 EOB Pattern and Mask Registers (04H to 07H)

The description of these registers is the same as for registers 00H to 03H, except that registers 04H to 07H control the recognition of an EOB condition.

5.3 Put BOB Address and BOB Word Flag Registers (08H to 0B)

With these registers, three flags can be set on the data stream. Any 32-bit word, which is sent after a BOB (in the range 0 to 30) can be flagged. Bits 4 to 0 in the registers define which word is to be flagged. Flagged words are copied into the Paged FIFO (section 4.2.2) that can be read by the SHARC processor. Bit 5 in each registers enables (bit 5 = '1') or disables (bit 5 = '0') the flag. In previous versions of the design of the FPGA a normal FIFO in stead of the current Paged FIFO was used and the number of words to be stored in it was depending on the value of this bit. In the current design this is no longer true, so that this bit can always be set to 1.

When a BOB is recognised in the data stream, the Header Counter is set to 0. The Header Counter is incremented on every 32-bit word in the data stream. When the word counter reaches the value 31, it stops counting and remains at this value until the next BOB is encountered. The counter value is compared to a value loaded in one of the registers 09H to 0BH. A 32-bit word in the data stream will be copied to the paged FIFO when the loaded value is equal to the word counter value and the flag is enabled (bit 5). Note that register setting *0000001FH is illegal!* This is because the word counter stops incrementing and keeps its value when it reaches 31. The first word after a BOB in the data stream is marked as word 1. To copy this word the register needs to be loaded with 00000021H. If a register is filled with 00000020H then the BOB word itself is copied. However, the BOB is already copied by default so this register setting is redundant.

Figure 2 and 3 illustrate the selection of words that should be copied to the paged FIFO for later use by the SHARC. In fig. 2 the fourth word after the BOB is selected, by programming register 09H with a value of 24H (Note that writing 04H to the register would result in setting the selection flag for the fourth word but it would not be copied because of bit 5 being '0').

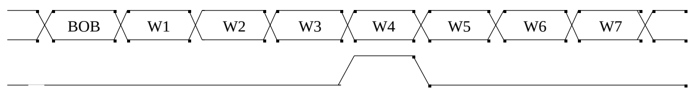

Fig. 2 Word 4 is flagged

Register 08H has a special function. It specifies when the buffer address of the BOB word is written to the paged memory. Fig. 4 and 5 illustrate why this is necessary: the words flagged are copied into the paged FIFO when they occur in the input data stream for the FPGA. In fig. 4, word W1 is directly following a BOB word. Suppose word W1 does not need to be copied to the paged FIFO, while word W4 should be copied. Word W1 will not be selected by one of the BOB word flags 0 to 2, but word W4 will: during input of word W1 into the FPGA the buffer address (that is, where the BOB word is stored in the Buffer Memory) can be written into the Paged FIFO.

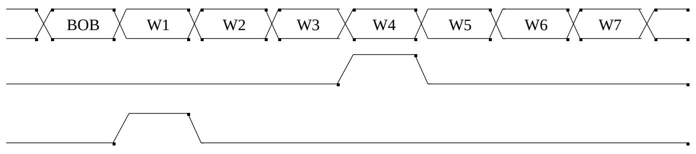

Fig. 3 W4 should be copied into the paged FIFO, the BOB address can be written into the Paged FIFO during W1.

On the other hand, if word W1 should be copied then one of the BOB word flags 0 to 2 has to flag it and the BOB address must be written later. In example 4 word W2 does not need to be copied so the BOB address can be written at that time.

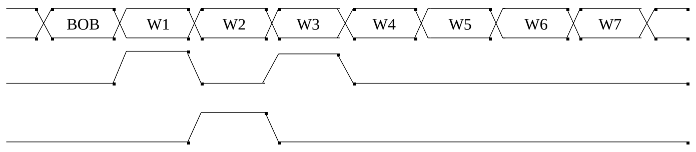

Fig. 4 W1 and should be copied into the paged FIFO, the BOB address can now be written into the Paged FIFO during W2.

5.4 EOB Word Flag Registers (0CH to 0DH)

By means of these registers, the two 32-bit words just before an EOB can be flagged. Bit 0 in the registers defines whether or not the flag is enabled. If bit 0 = '1' then the flag is enabled, if bit 0 = '0' the flag is disabled. Register OCH flags the second 32-bit word before an EOB. Register 0DH flags the 32-bit word just before an EOB. Fig. 6 shows how register 0CH and ODH flag the data stream.

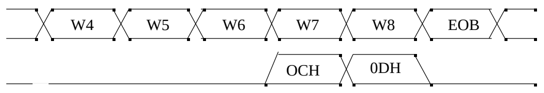

Fig. 5 EOB Word Flags (register OCH and ODH)

5.5 Maximum Event Size Register (0EH)

The value in this register determines the maximum allowable event fragment size. If the number of words in an event fragment (exclusive the BOB word) is more than the value in the register, the event is truncated. This means that no more words are written into the Buffer Memory. An “Event Overflow” interrupt may be generated to signal the SHARC that something went wrong (see Register 0FH).

5.6 IRQ0 Control Register (0FH)

This register controls the interrupt output IRQ0_n to the SHARC. There are three possible interrupt sources. If register 0FH is read back then bits 0 to 2 determine which of the three interrupt sources generated the interrupt:

- Bit 0: Paged FIFO Empty Interrupt,
- Bit 1: Paged FIFO Full Interrupt,
- Bit 2: Event Overflow Interrupt.

Writing a ‘1’ to one of these bits in register 0FH clears the interrupt.

Bits 3 to 5 are the mask bits for the interrupt bits (bits 0 to 2). An interrupt bit can only be set when the corresponding mask bit is a ‘1’ :

- Bit 3: Paged FIFO Empty Interrupt Mask. If ‘1’ then a FIFO Empty condition generates an interrupt,
- Bit 4: Paged FIFO Full Interrupt Mask. If ‘1’ then a FIFO Full condition generates an interrupt,
- Bit 5: Event Overflow Interrupt Mask. If ‘1’ then an interrupt is generated if the Event contains more data words than is programmed in the “Maximum Event Size” register 0EH.

5.7 IRQ1 Control Register (10H)

This register controls the interrupt output IRQ1_n to the SHARC. Reading this register after an IRQ1_n occurred, yields a Buffer Memory address (bits 17 down to 0). On this Buffer Memory address, a word is stored which contains an error indicated by the LDERR_n bit of the S-Link. If bit 18 of this register is a ‘1’ then there was an overrun condition. This means that one or more previous errors were not read out in time and are overwritten with the Buffer Memory address of the last error. Writing any value to this register clears IRQ1_n.

5.8 IRQ2 Control Register (11H)

This register controls the interrupt output IRQ2_n to the SHARC. This interrupt becomes active when two successive BOB patterns or two successive EOB patterns were found in the link data (a double BOB or EOB condition). Reading the register after an IRQ2_n occurred yields the Buffer Memory address on which the second BOB or EOB word is stored. A double BOB or EOB condition can occur in the case that the data on the link is corrupted or if an event is shortened because its size was bigger then the maximum event size programmed (in register 0EH). If bit 18 of this register is a ‘1’ then there was an overrun condition. This means that one or more previous errors were not read out in time and are overwritten with the Buffer Memory address of the last error. Writing any value to this register clears IRQ2_n.

5.9 Test Registers (12H to 14H)

These registers are used in 'Test Mode'. In test mode, the S-Link input is disabled, and is taken over by the SHARC. The SHARC can now write data as if it came from the S-Link.

Register 12H is the test control register. This register contains the following bits:

- Bit 0: if in test mode, taking this bit from 0 to 1 writes test data in register 13H and 14H,
- Bit 1: '0' Normal operation (S-Link), '1' test mode operation.

Register 13H should be programmed with the 32 data bits which would normally come from the S-Link LD0-LD31 bus.

Register 14H contains the following bits:

- Bit 0: status of the LCNTR_n bit that would normally come from the S-Link,
- Bit 1: status of the LDERR_n bit that would normally come from the S-Link.

After power-up, the CRUSH operates normal (that is, no test mode). To run a test, write the following data in the corresponding registers:

```
12H:  00000002H   (Enter the test mode)
13H:  S-Link Data word 1
14H:  S-Link Corresponding status LCNTR and LDERR
12H:  00000003H   (Write Data Word 1 by taking bit 0 from 0 to 1)
12H:  00000002H
13H:  S-Link Data word 2
14H:  S-Link Corresponding status LCNTR and LDERR
12H:  00000003H   (Write Data Word 2 by taking bit 0 from 0 to 1)
12H:  00000002H
      .
      .
      .
13H:  S-Link Data word N
14H:  S-Link Corresponding status LCNTR and LDERR
12H:  00000003H   (Write Data Word N by taking bit 0 from 0 to 1)
12H:  00000000H   (Go to normal operation, exit test mode)
```

6. Data format in the Paged FIFO

The Paged FIFO contains pages of 8 words. For readout the Paged FIFO behaves as a normal FIFO. The data read out is organized in 8 word groups, as indicated in table 2.

Word	Contents
0	BOB Pattern
1	Buffer Memory address where the BOB word is stored
2	Data flagged by BOB Word Flag 0 (if enabled)
3	Data flagged by BOB Word Flag 1 (if enabled)
4	Data flagged by BOB Word Flag 2 (if enabled)
5	Data flagged by EOB Word Flag 0 (if enabled)
6	Data flagged by EOB Word Flag 1 (if enabled)
7	EOB Pattern

Table 2 Contents of a group of 8 words, as supplied by the Paged FIFO

If one or more words from Word 2 - 6 are not enabled, the corresponding word(s) in de page will contain an undefined value. In the case of a double BOB condition (see register 11H), the words will contain data that is valid for the last block of data (in between a BOB pattern and an EOB pattern) which was received on the link. The data of previously received blocks (which were not terminated by an EOB pattern) are overwritten. In the case of a double EOB condition (see register 11H), the page will contain corrupted data since there was no BOB pattern on the link that matches the last received EOB pattern.

7. Reset facilities

There are three reset signals: “GlobalRst_n”, a “Rst_n” and “LRst_n”.

The GlobalRst_n signal becomes active during a power-up of the CRUSH, during a “Manual Reset” (reset button), or an “External Reset” (2-pin header). The GlobalRst_n signal resets the SHARC, the FPGA (including the Pipeline, Address Generator and the Paged FIFO). Note that the S-Link and the S-Link FIFO are not reset.

The “LRst_n” signal resets the S-Link and the S-Link FIFO. The S-Link will force LDOWN# low until the initialisation phase of the S-Link is complete. LDOWN# will then go high again. The S-Link is now up and running again. “LRst_n” can be controlled by the SHARC using flag0.

The “Rst_n” signal is used to reset the whole FPGA (including the Pipeline, Address Generator and the Paged FIFO). “Rst_n” is active during a GlobalRst_n, and can be controlled by the SHARC using flag1.

The following sequence should give a proper initialisation. First there will be a global reset. After that, the SHARC is booted through one of its links. In the mean time garbage data has a chance to enter the S-Link FIFO and the FPGA. Therefore the SHARC should activate flags 1 (Rst_n) and flag 2 (Freeze_n). After at least one clock pulse the Rst_n signal can be de-activated. The FPGA is now reset and no (garbage) data can flow into the FPGA because of the Freeze_n signal which is still active. The S-Link reset state machine in the FPGA is now able to reset the S-Link and the S-Link Fifo. This is done by activating flag 0 (LRst_n). After the initialisation phase of the S-Link, data may flow into the S-Link Fifo. The Freeze_n signal can now be de-activated so the S-Link data can flow into the cleared pipeline of the FPGA

8. The flag inputs and outputs of the SHARC

The SHARC has 4 flag signals, which can be configured as input or output signals under software control. Flag 3 is used as input, a “1” indicates that the Paged FIFO is Empty. The other three flags are configured as output:

- flag 0 is connected to the “LRst_n” signal that resets the S-Link and the S-Link FIFO,
- flag 1 is connected to the “Rst_n” signal that resets the FPGA (including the Pipeline, Address Generator and the Paged FIFO),
- flag 2 is connected to the Freeze_n signal. When asserted this signal stops the data flowing from the S-link FIFO into the FPGA

9. Address spaces of the SHARC

Three different address spaces are used:

- MS0 for read / write access of the Control and Status Registers of the FPGA,
- MS1 for read access of the paged FIFO of the FPGA,
- MS2 for read / write access of the ZBT Buffer Memory.