FE SLINK-MFCC status

Authors : SV, DF, MJ, GM, JP
Keywords :

Abstract

Current understanding of the MFCC usage as an input to an Slink board.

NoteNumber :
Version :
Date : 2nd September 1999
Reference :

1 Scope

This document is intended as an aid to the discussion on the MFCC implementation of the ATLAS ROBIN. It should be used to generate an MFCC-ROBIN specification document.

2 Status of the FE part of the MFCC

2.1 July status

During previous months an early study on the possibility of using the MFCC as a destination card for the SLINK receiver board has been done. A sketch of the MFCC board is shown in Figure 1.

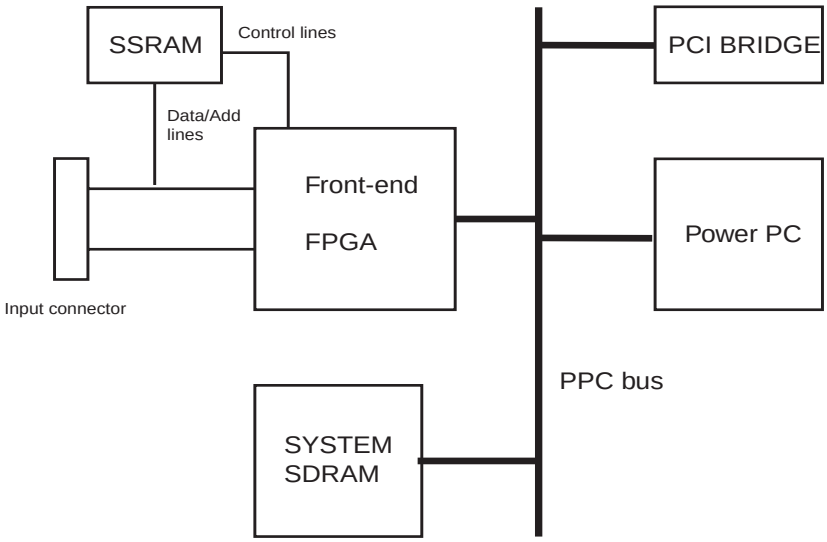


Figure 1: MFCC block diagram (current release, only the blocks relevant to FE, the connection of the FE-FPGA to P2 is not shown).

The Slink protocol requires that data is transferred out of the SLINK as if the Local Destination Card (LDC) would be writing into a FIFO, i.e. the LDC provides data synchronized with its own clock, LCLK.

The MFCC FrontEnd (FE) FPGA has some signal lines directly connected to the FE connector, some lines shared with the SSRAM address and data lines. It also drives all SSRAM control signals. The SSRAM is single-ported. For these reasons (it could create a serious bottleneck in the data path) the possibility of storing LDC data in the SSRAM has been ruled out.

The study then focussed on the possibility of having slink data stored directly into the MFCC System DRAM, either having data pushed by the FE FPGA (PPC master and slave interfaces required) or having data read and written by the PPC processor (PPC slave interface required).

The MFCC design, as it was presented some months ago, required an additional FIFO between the LDC and the FE FPGA and the Optional SSRAM not mounted/used. The solution presented in July was the one shown in figure 2.

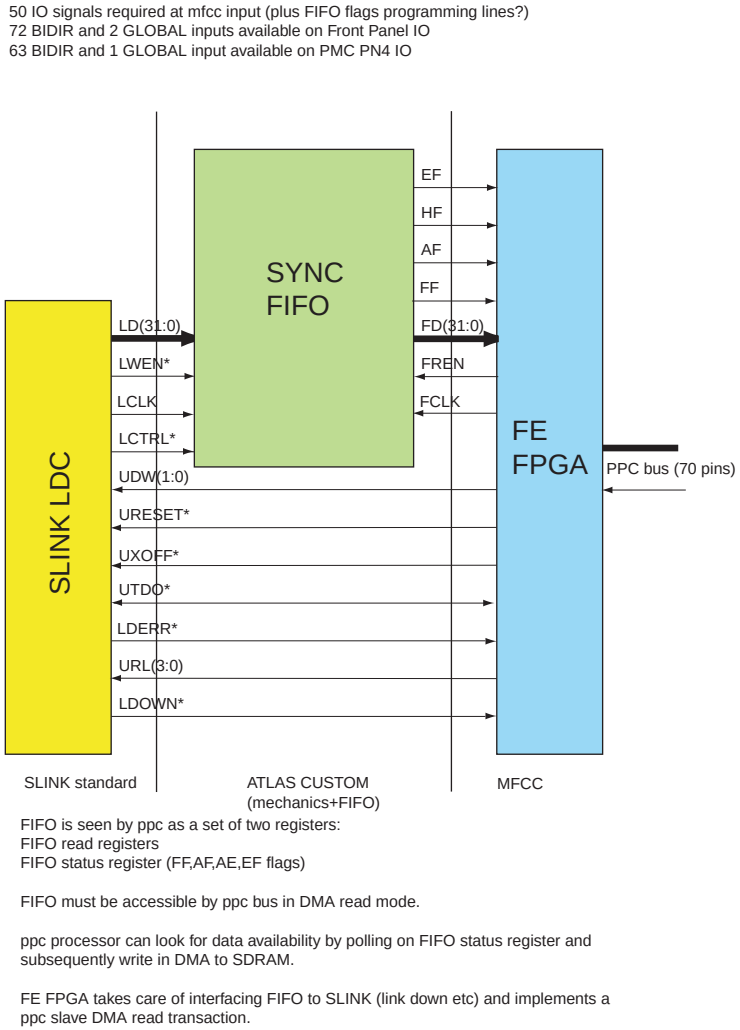


Figure 2: July solution for interfacing LDC to MFCC.

The double clocked FIFO was needed in front of the FE FPGA to resynchronize SLINK data to the rest of the MFCC board. At this idea CES suggested to implement the FIFO directly into the FPGA.

The data could be trasferred by polling a register accessible from the ppc bus containing the fifo status flags.Status (990113)

Last contacts with CES have given new informations, much work as been done on their side driven by suggestions from other customers:

1) The current development of the FE FPGA foresees:

- a PPC Slave interface, capable of single RW accesses and of reading in cached line mode.
- a set of user definable registers accessible in RW.
- a FIFO (32 bit width x 128 depth), implemented with a cycle shared RAM. This FIFO can unfortunately be operated only at 16 MHz maximum. SLINK requires 32 bit data + 1 Control bit and 33 MHz operation.

2) The board will mount a new version of the Flex 10k FE FPGA, with improved speed and Double Ported RAMs, capable of implementing a Double Clocked FIFO, as required in our design.

3) Next version of MFCC, coming out by end of February will have also the SSRAM completely decoupled by the FE lines, for a better usage of this memory, if needed.

Very preliminary studies, done with tools available at CERN, indicate that a 33bit wide X 128 bit deep fifo in this technology could run at about 50 MHz (we need 40 MHz max), using up about 10% of memory space and 5% of logic space of the FPGA.

CES has released some new documentation.

CES is willing to modify the PPC slave, in order to be able to read at possibly 33 MHz.

CES states that it is not possible to implement a ppc master capable of pushing data into the SDRAM because it would consume much or all of the currently used FE FPGA resources.

One possible solution would be the one shown in figure 3.

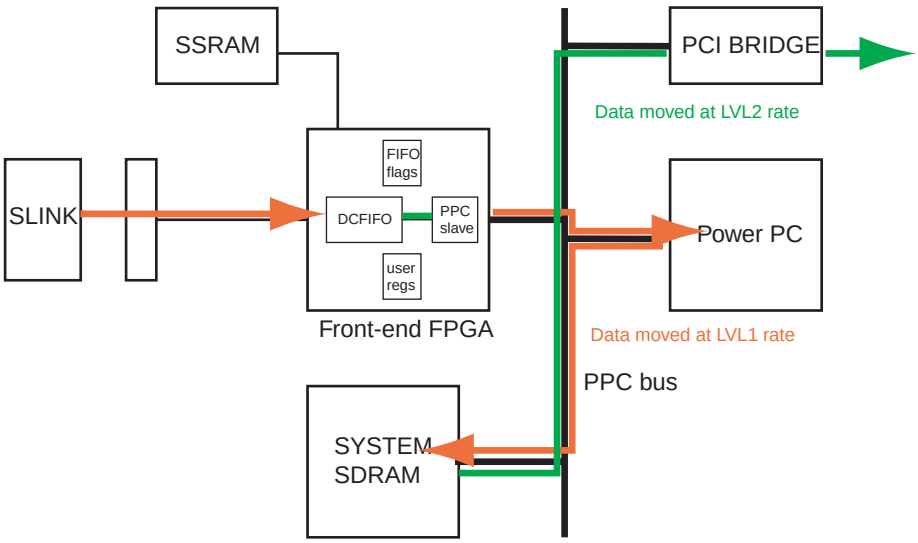


Figure 3: New MFCC version and possible data flow.

To be discussed

Is the current approach valid ?

If the approach would be considered some valid actions could be taken:

1. Current CES development could be evaluated, to test data transfers in cashed line mode from FE-FPGA to SDRAM and measure CPU usage (the single cycle RW access on SDRAM of 70/80 MB/s is of some concern). The hardware required to download FPGA is not available at the moment. Code sources are available to generate download files.

2. We could define and simulate the solution presented in Figure 3, with the next generation device which will be mounted on new MFCC. All software tools are available at CERN. MFCC for evaluation will arrive end of February.
3. Understand the electrical connection and timing properties of the available SLINK to MFCC adaptor, if not yet done.
4. Would it be better technically to consider input to the MFCC via the P2 connector?
5. Do we need a 6U board to hold FC technology (LDC)?
6. Could we ask CES to use a bigger FE-FPGA to implement the ppc master?

2.2 Status (990120)

Current simulation of the FE-FPGA now includes the DCFIFO (in CES test-bench). Status flags are accessible by a user register. The ppc-slave block has been adapted to accept real FIFO flags to properly terminate cached line reading (currently fifo control logic is inside ppc-slave, it is not capable of handling external data).

The final DCFIFO fitting code has to modified by CES, to check the real implementation.

CES should modify the ppc-slave interface to read FIFO at full speed, if needed.

At simulation level the CES test-bench has to be modified to allow for external data.

The SLINK protocol has to be implemented. The basic idea is the following:

- implement 32 bit word transmission (8 and 16 bit data transfer not foreseen in ATLAS, loss of a factor 4 or 2 on the output bandwidth).
- Reset signal driven by URESET flag in user register
- flow control compares number of data words into fifo with NWORDS field in user register. Transmit off signal generated when number of fifo words equal or greater that NWORDS.
- link transmit error is monitored by user register flag XERR. It could also be written along with data (word by word error reporting).
- Control word flag written along with normal data into FIFO.

All the hardware required for testing is available. Code sources have been compiled and the downloading files are ready. Software available only for single accesses now but cached line reading functions should come in about one week.

Now two possible implementation are under study:

1) master ppc interface and buffer management at FPGA level. Master ppc requirements and implementation issues have to be discussed with CES. In the current architecture the front-end FPGA is not equipped with a PPC master. Therefore, the only way to move data from it to a buffer SDRAM is to use the CPU. This, however, is not very efficient since the data unnecessarily has to pass through the CPU and hence is transferred twice across the PPC bus (FPGA->CPU and CPU->SDRAM).

As CES explained to us it would be very difficult to put a general purpose PPC master and DMA controller into the front-end FPGA. This is mainly because support for cache coherency and crossing of address boundaries would cost a lot of gates. In our case, however, where the events are small, stored in aligned buffers and not (except for the header) accessed by the CPU one can make some simplifications:

- The memory reserved for event data storage will be marked non-cacheable and the PPC master will not check for cache inconsistencies

- Each transfer from the front-end FPGA to SDRAM will start at a 1KByte aligned address

- DMA transfers are not allowed to cross certain address boundaries (8 KBytes, if I remember right)

According to CES, these simplifications would allow them to shrink the PPC master / DMA controller to an acceptable size.

What is less clear right now is how the buffer management would have tobe done. Currently the Event Manager maintains a list (software FIFO in SDRAM) of free pages (data buffers). Each time a new event arrives one of these pages is allocated and removed from that list. If the front-end FPGA is supposed to initiate the event transfer by itself, it would have to have access to this list. A fraction of this list could be written (limited by the amount of space available) into the front-end FPGA, where it would be accessed locally. The details of this issue have to be discussed with CES.

2) a slave ppc interface and buffer management at ppc Processor level. This solution uses more heavily the ppc bus and processor but is simple and requires small efforts from CES. Still remains to be decided if this implementation should be done and its performance measuremed.

A short discussion with Van Der Bij revealed that no electrical test has been done on P2 connector, long connection stubs could be dangerous (need to undestand responsibabilities of that group). Current adaptor uses P2. A study has to be done whether the front connector can be used (mapping is understood, it has to be checked by CES).

Clearly a formal meeting/request to CES should follow as soon that our needs are clear.

2.3 Status (990125)

The simulation of the ppc-slave is almost complete, it now includes the slink interface, with flow control. It is possible to monitor the link status from a user register. It now needs validation from CES on the following points:

- update our modified ppc slave, to handle 34 bit fifo at full speed, and include new CES revision received last week.
- top level block has to accomodate new IO pin names and dcfifo.

The master-ppc mode of operation has been defined. It is shown in Figure 4.

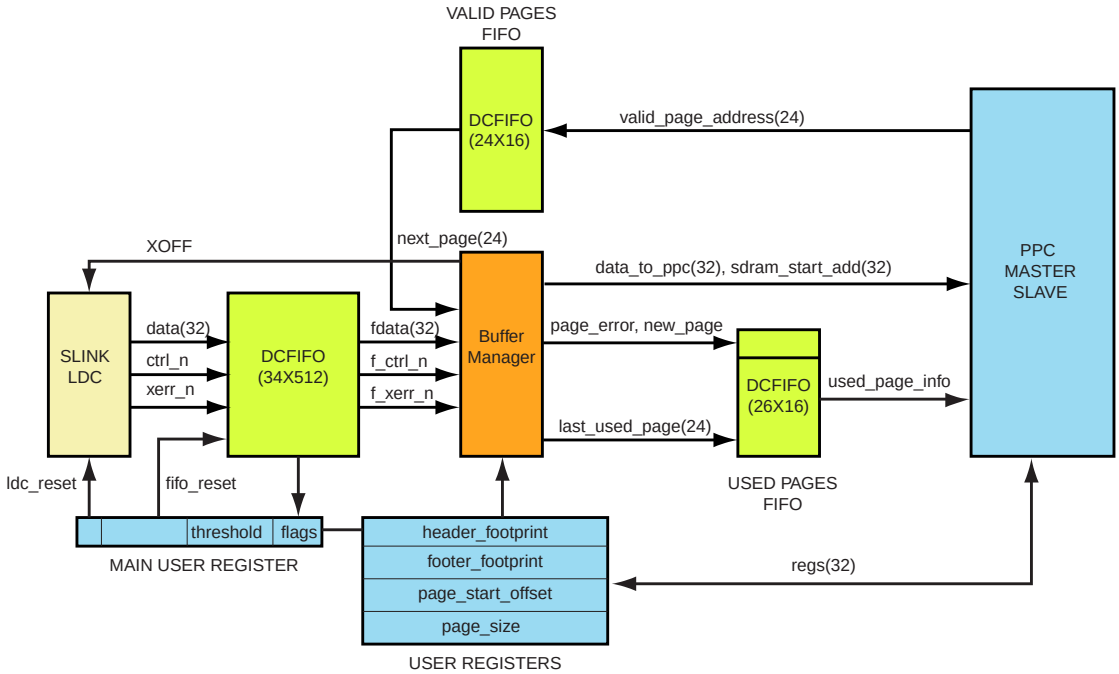


Figure 4: Master-ppc FE-FPGA block diagram

1. The Slink LDC sends 32 bit of data, control bit and error bit to a 34-bit wide 256-words deep FIFO.
2. LDC clock drives the FIFO Write Clock (Wrclk).
3. LDC Reset is given by the content of the ldc_reset bit in the MAIN user register.
4. Each FIFO has its own reset bit in a user register.
5. Transmit OFF (XOFF) is generated by comparing a threshold field in the MAIN with the FIFO filling state (Rrusedwords).
6. FIFO reset is given by the content of the fifo_reset bit in the MAIN user register.
7. FIFO flags (Rempty, Rfull, Rrusedwords) are monitored via the MAIN.
8. A state machine (buffer_manager) is in charge of the following:
   - check for new pages in the VALID_PAGES FIFO;
   - check for data in DCFIFO;
   - find packet header (control word with expected header_footprint);
   - send PPC MASTER interface DMA starting address, with proper page_start_offset;
   - read FIFO, send data and count number of words;
   - check whether number of words exceeds page_size;
   - check for packet footer (control word with expected footer_footprint) or for end of page;
   - the number of pages per event will be limited to a value written into a user register.

- at the end of packet or at end of page or having reached the maximum number of pages, write page_error (if an error flag was set during the current packet) or new_page (new_page is set to FALSE if the packet is not finished yet) to USED_PAGES FIFO, together with the used page address.

9. A set of USER register:
    1. MAIN (already described)
    2. header_footprint (32 bit)
    3. footer_footprint (32 bit)
    4. page_start_offset (16 bit)
    5. page_size (16 bit).
    6. max pages (8 bit).
10. a DATA FIFO (dcfifo) 34bit wide by 256 words deep.
11. a VALID PAGES FIFO (24 bit wide by 16 words deep)
12. a USED PAGES FIFO (26 bit wide by 16 words deep).
13. a PPC MASTER, capable of generate DMA write cycles to the SDRAM.
14. a PPC SLAVE, capable of RW in single access the user registers, READ the DATA FIFO, RW VALID PAGES FIFO, RW USED PAGES FIFO.

A decision on register and FIFO sizes has to be taken. The numbers shown are only indicative.

2.4 Status (990201)

The fe-fpga with ppc slave interface has been developped on the existing MFCC version. It has a small FIFO, 34 bit wide, 16 word deep, the only that can be implemented without any available DPRAM blocks. The new program has been downloaded into MFCC with the SLINK protocol.

Some issues on error correction in the buffer_manager have been defined:

- if control word does not contain the header_footprint, the buffer_manager issues a used page with the error flag set and reads the data fifo until it finds a good header or until the it is empty.
- if a control word is found without a trailer footprint, the buffer_manager closes a page, with the error flag set.
- if the maximum number of pages is reached, the buffer_manager closes last page and issues an error, then starts looking for a new header.
- if the buffer_manager finds data into the fifo and no header, it flushes the fifo until it finds a new header.
- if the buffer_manager finds a header, then a data packet, then a new header, it closes the page with the error flag set.

2.5 Status(990210)

The S2P2, Slink to MFCC interface has been tested, with a SLIDAS generating a test pattern, running at 40 MHz. Pin assignment is correct. It looks fine.

A small test program has been run to check the functionality and the transmit-off logic. It works fine when the FIFO threshold is set to two words, i.e. the XOFF_N signal has to be issued 14 clock ticks before the FIFO fills.

All functionalities of the ppc-slave option have been implemented. A map of the two user registers follows:

ADD 0x	user_reg_0 bits
lemoen_n	*26*
groupb_in_n	*25*
groupa_in_n	*24*
slink_ureset_n	*23*
lemb[3..0]	*15..12*
lema[3..0]	*11..8*
fifo_thresh[6..0]	*6..0*

Table 1: user register 0, RW access

ADD 4x	user_reg_1 bits
fifo_error_n	*28*
fifo_ctrl_n	*24*
fifo_empty	*16*
fifo_full	*12*
no_words_in_fifo	*7..0*

Table 2: user register 1, R access

A rough evaluation of bandwidth requirements has shown that the only viable solution is to use the ppc-master option. Given 100 KHz LVL1 trigger rate and an average event size of 1 kB, we expect 100 MB/s (BW) from the SLINK.

The ppc-slave option would require 2 BW for transferring data from MFCC to PPC processor, BW from the processor to DRAM, and approximately 0.5 BW for other needs. In total the required bandwidth on the ppc bus would be 350 MB/s.

It has been decided that the ppc-slave option is a good start but focus has to be moved to the final implementation:

- it has permitted to undestand the software tools (all is available as central resources at CERN).
- the SLINK protocol has been implemented and tested
- the electrical adapter S2P2 has been tested.

The ppc-master option requires only BW to move data from MFCC to DRAM and 0.5 BW for other needs (ROI requests ...), for a total of 150 MB/s (100 MB/s would be cached line access).

A baseline solution for the ATLAS ROBIN has been defined now. It requires input from CES on time schedules. Also a clear partition between ATLAS specific and CES general facilities is needed, to work efficiently and in parallel on the new MFCC.

2.6 Status(990601)

A first ROBIN logic scheme has been simulated, in order to be ready when CES PPC master block will arrive.

The schematics is shown in Fig.5, (a detailed description of each block and its connectivity will be written soon).

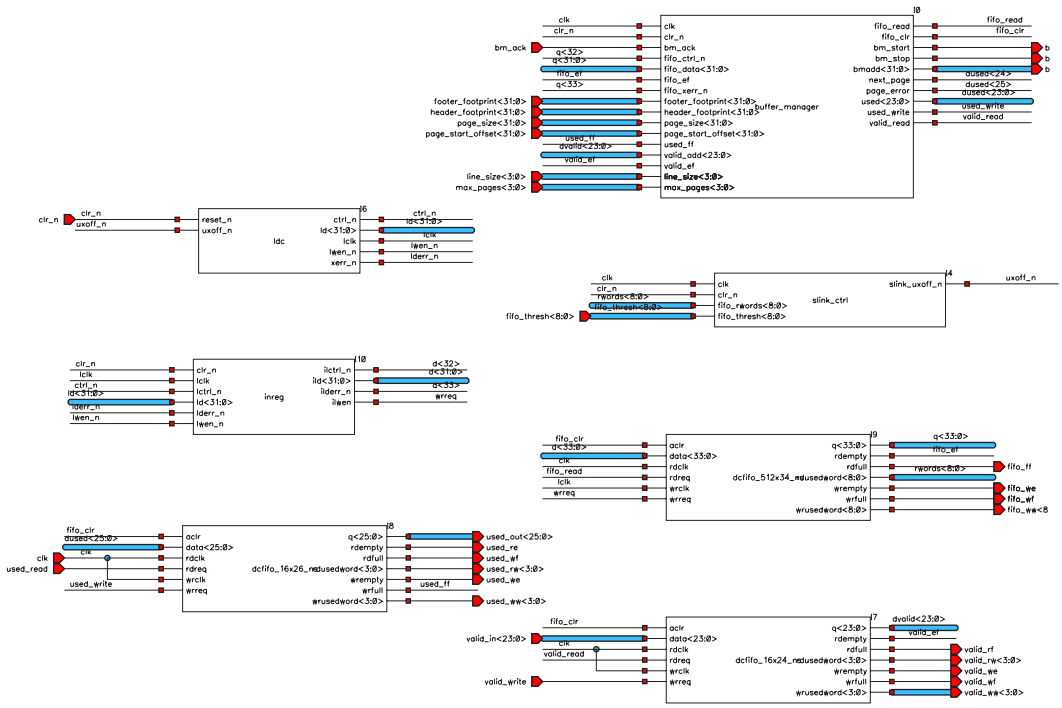


Figure 5: ROBIN block schematic

The buffer_manager state machine design is debugged, but some assumptions are done on the PPC-master block. Block transfer is currently completely controlled by the buffer_manager, this will probably differ from the final implementation, where complete control might not be

available. A bubble diagram of the buffer_manager states is shown in Fig.6. Further optimization of the buffer_manager block will be done when all the design blocks are available.

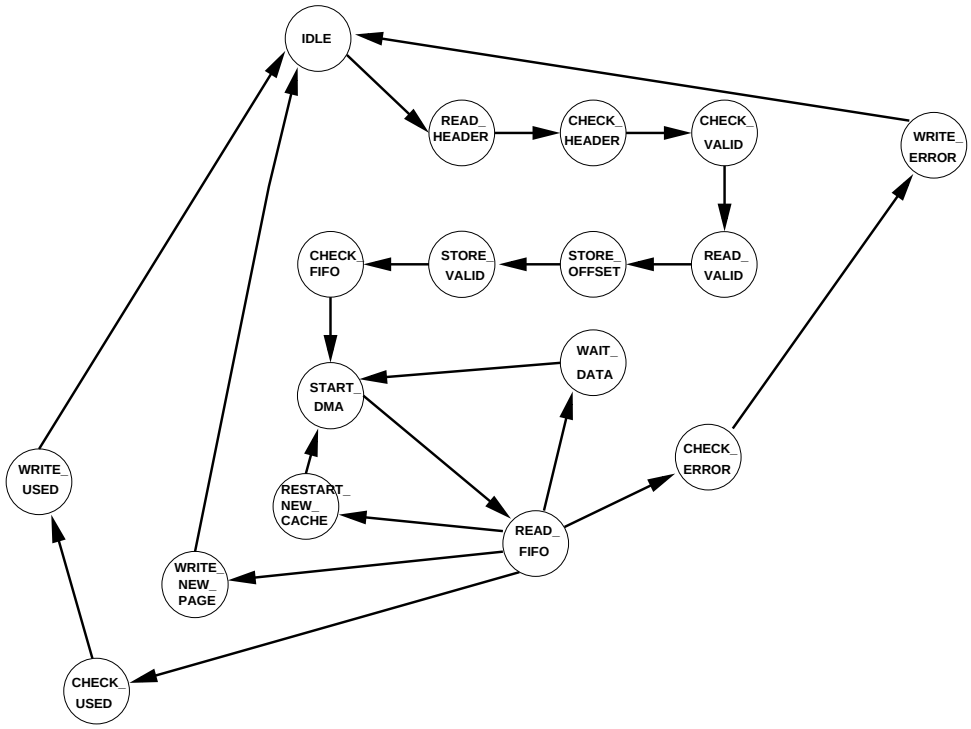


Figure 6: buffer_manager state machine block diagram.

Simulation work has been done to debug and evaluate the state machine design and its input-output signals requirements. An example is shown in Fig.7, where a complete event coming in the ROBIN is written in two memory pages.

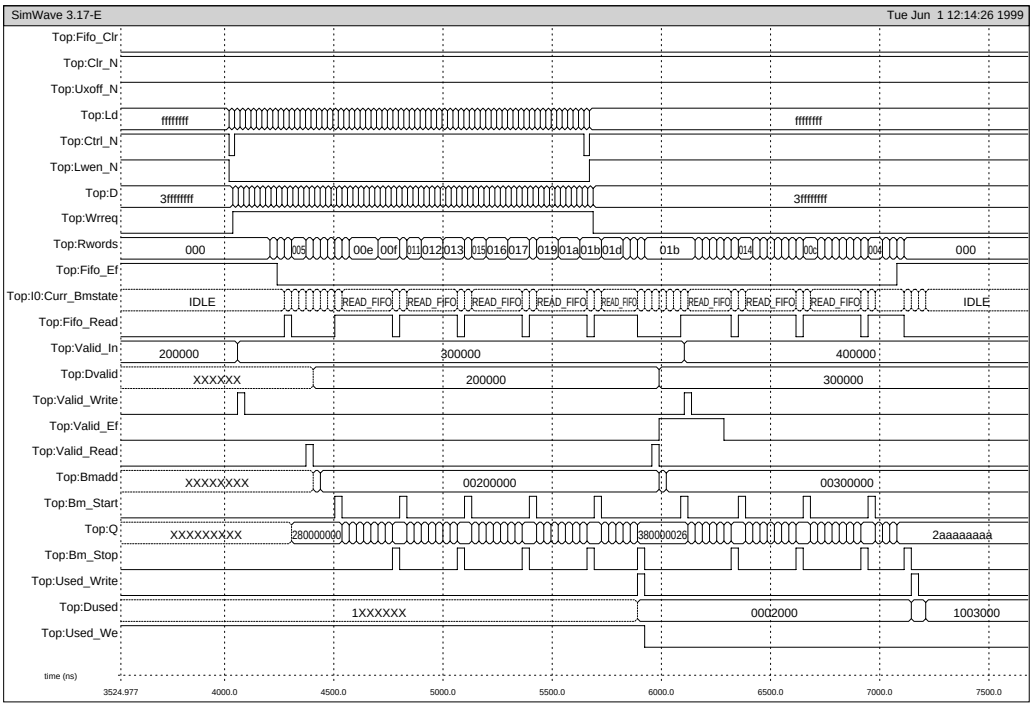


Figure 7: Handling of an event spanning over two memory pages (page sizes and event size values used are just demonstrating the capability of the state machine and are not the final ones).

A list of the required control and status register follows. The base addresses could be subject to changes in the final implementation.

Table 3: Footer register: this register contains the ATLAS footer footprint

ADD 00x	footer_reg bits
footer_footprint(16 MSBs)	*15:0*

Table 4: Header register: this register contains the ATLAS header footprint

ADD 04x	header_reg bits
header_footprint (16 MSBs)	*15:0*

Table 5: Page control register: this register controls the pages sizes, offsets, and the maximum number of pages to be written for the same event, before issuing an error condition.

ADD 08x	page_control_reg bits
max_pages	*31:28*
page_start_offset	*23:16*
page_size	*15:0*

Table 6: Valid pages FIFO: it contains up to 16 valid page addresses, to be used by the ROBIN

ADD 0Cx	valid_pages_reg bits
valid_add	*15:0*

Table 7: Used pages FIFO: it contains up to 16 page addresses, used by the ROBIN

ADD 10x	used_pages_reg bits
page_error	*31 (R only)*
next_page	*30 (R only)*
used_wcnt	*23:16 (R only)*
used_add	*15:0 (R only)*

Table 8: Main status register: it contains the main data fifo flags and slink and buffer manager status

ADD 14x	main_status_reg bits (R only)
slink_ctrl_n	*31*
slink_error_n	*30*
slink_down_n	*29*
buffer_manager_busy	*28*
valid_ef	*25*
valid_ff	*24*
valid_words	*23:18*
used_ef	*17*
used_ff	*16*
used_words	*15:10*
main_fifo_ef	*9*
main_fifo_ff	*8*
main_fifo_words	*6:0*

Table 9: Main Control register: it contains the maximum DMA transfer size and the slink interface registers.

ADD 18x	main_control_reg bits
page_hi_offset	*31:24*
dma_fifo_clear	*22*
ppc_fifo_clear	*21*

ADD 18x	main_control_reg bits
used_fifo_clear	*20*
valid_fifo_clear	*19*
main_fifo_clear	*18*
buffer_manager_clear	*17*
dma_manager_clear	*16*
line_size (not used)	*15:12*
slink_clear_n	*11*
dma_manager_on	*10*
buffer_manager_on	*9*
main_fifo_threshold	*8:0*

Table 10: slink to P2 board control and status register

ADD 1cx	s2p2_reg bits
slink_ld(31:0)	*31:0*

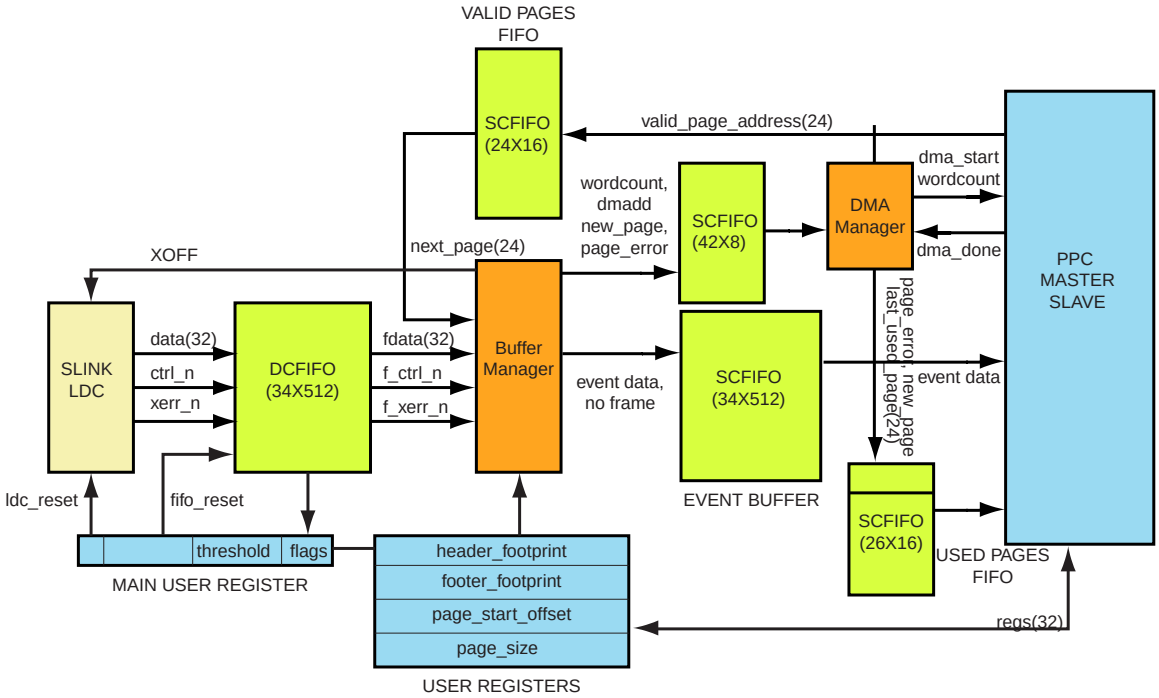
2.7 Status (990902)

A complete ROBIN scheme has been simulated and fitted into the new FPGA device. An additional buffering stage, controlled by a second state-machine called DMA-manager, has been added for the following reasons:

- DMA Master block needs a wordcount to initiate the DMA
- the DMA begins after a complete event or a full page is filled. In this case we have a better usage of the PPC bus bandwidth.

The communication between the buffer-manager and the DMA-manager is done via a DMA FIFO, which contains one full DMA address and wordcount per page.

Figure 8: New ROBIN scheme



The achievable speed is 60 MHz on the Slink clock, and 68 MHz on the 66 MHz clock. In order to achieve the speed some cleanup of unnecessary features has been done on the PPC master part.

On the ROBIN specific part now the Slink clock drives the Slink FIFO, the buffer_manager state machine, the reading of the Valid pages FIFO and the writing of the Event FIFO. The 66 MHz clock drives the PPC master and slave parts, the Robin internal registers, the writing of the Valid pages FIFO, the DMA_manager state machine, the internal DMA_manager FIFO, the Used pages FIFO.

Current FIFO sizes are as follows:

FIFO name	type	widthxdepth
Slink fifo	*Single-clock, memory-based*	*34x128*
Valid page FIFO	*Double-clock, Memory-based*	*16x64*
Used page FIFO	*Single-clock, memory-based*	*32x64*
PPC buffer FIFO	*Double-clock, memory-based*	*32x512*
DMA FIFO	*Double-clock, Flip-Flop-based*	*32x8*

Some FIFOs had to be implemented with Flip-Flops because of lack of resources in the FPGA.

A downloadable file is available for testing. The resources used are currently 91% of the logic and 51% of the total memory. In order to gain some resources a new addressing scheme has been envisaged. The implementation still does not contain this new addessing scheme:

- the addressing scheme should be changed, the USED and VALID pages FIFO will become 16 bit wide and contain 14 bit addresses, the middle prortion of the full page address. A page address used by the DMA manager will finally be composed like follows:

  PAGE_ADD[31:0]=PAGE_BASE[7:0] & ADD[13:0] & PAGE_START_OFFSET[7:0] & “00”

  where PAGE_BASE and PAGE_START_OFFSET will be contained in mfcc registers and ADD will be read(written) from(to) the Valid(used) page FIFO.

A to-do list follows:

- the Used pages FIFO is written by the DMA_manager with one word per page, containing the wordcount only. The page address has to be written as the second word.
- the PAGE_START_OFFSET is currently used in every page. It has to be used only in the first page of a multi-page event.

The tables in paragraph 2.6 have been updated.