The Module Controller Chip (MCC) of the ATLAS Pixel Detector

Giovanni Darbo[1], INFN-Genova

on behalf

ATLAS Pixel Collaboration [1]

Albany, Berkeley, Bonn, Dortmund, Irvine, Genova, Marseille, Milano, Nikhef, New Mexico, Oklahoma, Prague, Santa Cruz, Siegen, Toronto, Udine, Wisconsin, Wuppertal

Abstract

The ATLAS pixel detector is organized in 3 barrels and 5 forward and backward discs. The basic building block for each of those detector components is the detector module. There are a total of 2,228 modules, each one having 16 analog Front-End (FE) chips and a Module Controller Chip (MCC). Each module has 61,440 channels and must deal with a complex signal structure: 40 MHz event interaction rate, 75 kHz trigger rate and 2.5 μs trigger latency. The MCC has the main task to coordinate the 16 FE's: it does event building, handles errors and overflows and deals with trigger and synchronization signals. The ATLAS Pixel Collaboration is designing a radiation soft version of the detector module, as "demonstrator" of feasibility, before developing the final version.

This paper describes the MCC architecture and the prototype chip designed for the demostrator.

1. Introduction

The ATLAS pixel detector [2][3] is constituted of 3 barrel layers and of 5 forward and backward disks. Each barrel is organized into staves and each disk into sectors, both of which are in turn composed of modules. A total of 2,228 modules are used in the whole detector.

The read-out of the several thousand pixels hit amongst the $1.4 \cdot 10^8$ channels is a difficult problem.[4] To solve it, ATLAS has adopted a column-based read-out system because of its simplicity, and the high bandwidth operation provided by independent columns. High parallelism is maintained at each step of the read-out architecture together with apprpriate data compression. This is necessary to extract the extremely sparse information in the most efficient way. This read-out system is implemented using a tree-like structure with distributed intelligence at each node.[5]

2. Module

A perspective view and a simplified block diagram of the module are shown in Figure 1 and Figure 2. The two different blocks in Figure 2 are the *Front-End* (FE) chip, which is replicated 16 times, and the *Module Controller Chip* (MCC). The interconnections have been kept very simple, and all connections which are active during data-taking use low-voltage differential signalling (LVDS) standards to reduce EMI and balance current flows. Other signals use full-swing single-ended CMOS to reduce the pin count.

The interconnect topology between the MCC and the 16 FE chips in a module is a star topology using unidirectional serial links. This topology has been chosen to improve the tolerance of the system to individual component failure, as well as to improve the bandwidth by operating the serial links in parallel.

The ATLAS Pixel Collaboration is currently designing a "detector module demonstrator" which implements the final functionality required for the experiment, but with two main exceptions: it has slightly larger pixel cells (400 x 50 μm^2 instead of 300 x 50 μm^2), to leave adequate room for diagnostic and control logics, and it has electrical instead of optical inputs and outputs. In addition, the chips on the current built module are designed in radiation-soft technology.

[1] e-mail: darbo@genova.infn.it

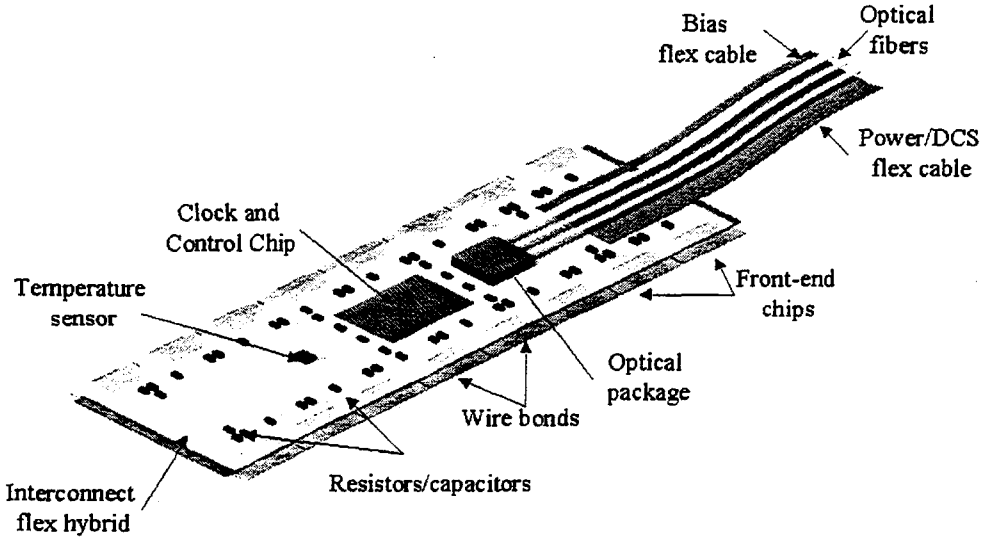

Figure 1: Perspective view of a detector module.

The prototyping effort for the demonstrator is being pursued with the aim of providing two front-end chip designs in rad-hard technologies using two different vendors, according to the official ATLAS policy for rad-hard electronics development. The two front-end designs (FE-A and FE-B) differ in many internal details, but are intended to be "functionally pin-to-pin compatible", so that modules can easily be built with either front-end chip and comparatively tested.

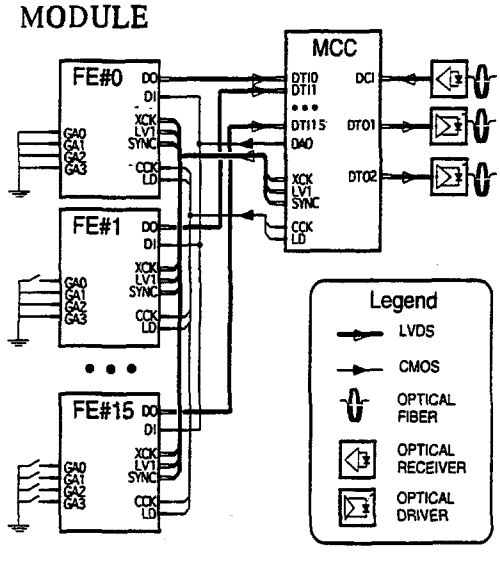

Figure 2: Module block diagram

Both FE chips have 18 columns (will be 24 with the pixels shrinking to 300 μm) and 160 rows of pixel cells. Every second column is mirrored and the architecture for the *End-of-Column* (EoC) logic is organized for a column pair. Both pixel front-ends will provide modest digital information using a time-over-threshold (TOT) front-end design, and digitizing the charge information in units of the 40 MHz beam crossing rate.

3. MCC System Architecture

The MCC has 3 main functions: event readout and building, FE chip configuration and trigger and timing distribution to the FE chips.

3.1. Event Readout and Event Building

The first significant event building occurs in the End-of-Column (EoC) logic on the Front End chip, where data are organized into events labelled by 4-bit trigger numbers. These events are then pushed from the FE chips to the MCC as soon as each FE chip has a complete event. The MCC collects the parallel data

streams from all of the FE chips and performs real event building. It then transmits these events to the ROD (Read Out Driver). The final event building tasks is left to the ROD's, where power and space are not as constrained as they are on the detector.

The End-of-Column logic receives input from a pair of pixel columns. All pixel hits are stored in the EoC buffers until a level one trigger (LV1) coincidence is performed. After that, only hits associated with a triggered beam crossing are kept and are transferred off the FE chip into the MCC. These basic operations, namely hit storage, LV1 trigger coincidence, and event readout, must be simultaneously performed by the EoC logic in order to prevent generating significant inefficiencies. Each FE chip uses a 4-bit trigger number to uniquely label events which are awaiting readout into the MCC. The MCC has the ability to suppress additional LV1 trigger signals if the number of LV1 triggers sent to the FE chips exceeds *n* (where *n* can be programmed from 1 to 15). This number represents the maximum number of events that a single FE chip can store.

The high luminosity of LHC forces an *"as-quickly-as-possible" approach in getting data* transferred from hit pixels to the End-of-Column buffers. This process causes data entering the EoC buffers to become somewhat scrambled, and will therefore no longer be organized sequentially into events. Hence this buffer will not behave as a FIFO, and data from a given event will not be stored contiguously. It becomes therefore necessary to scan the EoC buffer pool in order to find the next free location prior to writing new data. The buffer pool has also to be scanned a second time to find all the hits corresponding to a given event in order to provide the MCC with data ordered by event.

Event building is performed by two concurrent processes running in the MCC. The first (*Receiver*) deals with the filling of the 16 input FIFO's with data received from the corresponding FE chips, while the second one (*Event Builder*) extracts data from the FIFO's and builds up the events. Each FE chip sends data as soon as they are available with two constraints: event hits must be ordered by event number and for each event an `End-of-Event` (*EoE*) word is always generated. *EoE* is also sent for the case of an empty event to keep event synchronization.

Each time an *EoE* is received by a *Receiver* in the MCC this information is stored in a "score board". Therefore the *Event Builder* knows exactly when to start the event building by checking the "score board" which keeps track of which events are completely

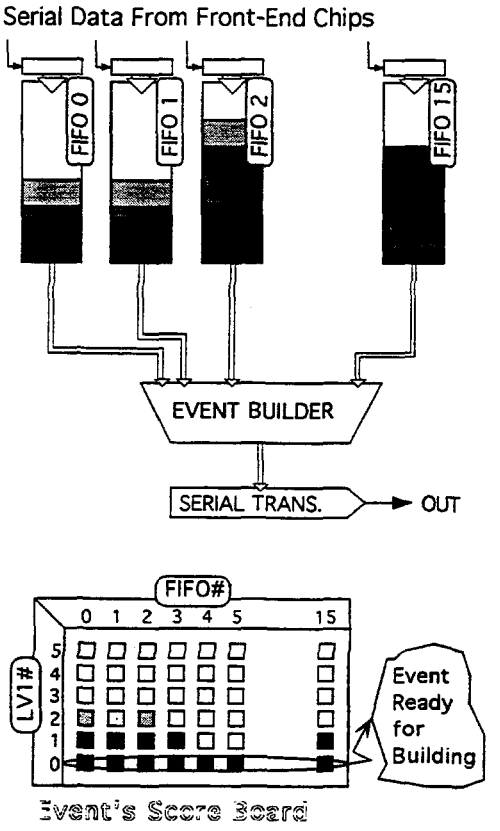

Figure 3: Event "Score Board" used in the MCC for event building.

stored in each of the 16 input FIFO's. When an event flagged with LV1 number *n* is ready in all the FIFO's the *Event Builder* process takes all the hits from FIFO#0, appends them to that of FIFO#1, and so on up to the last FIFO#15. At the end of this process the corresponding "score board" entry is erased. While the *Event Builder* fetches the hits out of the FIFO's, the MCC transmits them upstream to the *optical encoder*. The event number is removed from each hit and sent only once, together with the FE chip addresses, to the ROD. The event building score board is sketched in Figure 3.

The block diagram of Figure 4 is an expanded view of one of the sixteen FIFO's of Figure 3. During normal conditions, the RECEIVER fills data into the FIFO. When an *EoE* word arrives, CTRL copies the contents of W.PTR (Write Pointer) into L.PTR (Last Pointer). When the Event Builder finds that an event is completely received from all of the 16 FE chips

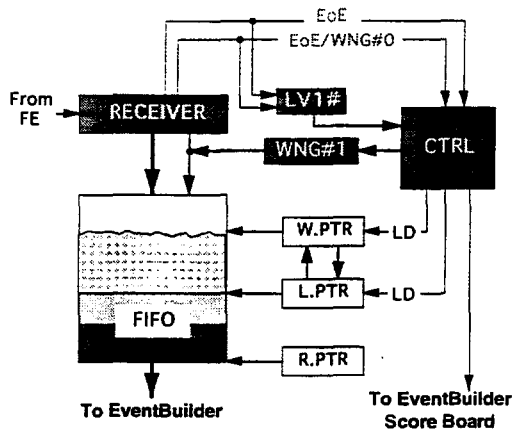

Figure 4: Event readout: Front End RECEIVER and input FIFO.

(*Event Builder* knows from the scoreboard about the existence of complete events) it starts fetching data. After every FIFO read operation its R.PTR is incremented. R.PTR will never overtake L.PTR and the *Event Builder* can start processing the next FIFO once it finds an *EoE* in the data.

The MCC is able to handle both warning and error conditions during event building. A `warning` occurs when there is a condition which causes a partial event loss. An `error` is a destructive condition that cause the loss of one or more events until the MCC recovers. Both `warning` and `error` conditions are flagged in the output data stream. Due to the data push architecture, the overflow of an input FIFO could always occur: the mechanism the MCC uses to resolve such a condition is to issue a `warning` or an `error` condition.

3.2. System Initialization and Configuration

Front end chips and MCCs must be configured after power up or before starting a data taking run. This is done by a system initialization procedure. To write or read configuration data to/from the FE chips, two levels of addressing are necessary:

1. The data path to the given module must be selected.This is done by selecting the I/O port on the ROD corresponding to the module on which the target FE chip resides. The ROD uses point-to-point links to the MCC's for both input and output connections.
2. The FE chip must be addressed. This is done by the FE chip itself, which recognizes its geographical address in the Command plus Address field of the message. Once the address is recognized, the remaining data stream will be used by the FE chip either to execute the command or to upload/download internal registers. Since all of the information going to the FE chips must pass through a MCC chip, the MCC must recognize whether the data is for its own use or must be transferred to the *MCC-DAO* (Data Address Out) pin of the FE port, which is connected to the FE chip inputs (see Figure 2). This is done by looking at the secondary address field present in the header field of the message.

Data streams sent to (or returned from) the FE chips have variable length. When writing, this length must fit the size of the relevant internal FE registers. To ease the FE chip design for the demonstrator chips, configuration data are transmitted at a reduced 5 MHz bit rate, using the *MCC-CCK* clock, instead of the 40 MHz used for event readout. Experience will tell us whether this can be eliminated in the next generation of chips.

To simplify the control decoder the *MCC-LD* signal is used to distinguish, in the bit stream from the *MCC-DAO* pin, the address plus control words from the subsequent data words.

The MCC architecture foresees up to 16 general registers which can be written and read back during system configuration. Two of them are used to define the length, respectively of the `data` and of the `control` part, in the bit stream which will be sent to the FE chip. The MCC registers support both system control functions and access status information that is particularly useful at debugging time.

3.3. Trigger, Timing and Control

During normal operation, the on-detector electronics of the pixel system needs a precise timing signal (XCK - Bunch Crossing Clock) and a trigger signal (LV1 - Level 1 Trigger). In addition to those two signals several control commands are understood by the system. The main signals are:

- XCK (Bunch crossing clock)

  This is the clock seen by the FE chip on its *FE-XCK* input line. XCK is generated from the 40 MHz clock signal received by the MCC through the *DCCI* input pin. The 40 MHz clock signal and *FE-XCK* both have the same frequency as the LHC machine clock. Each detector module uses the same 40 MHz clock, but a small phase

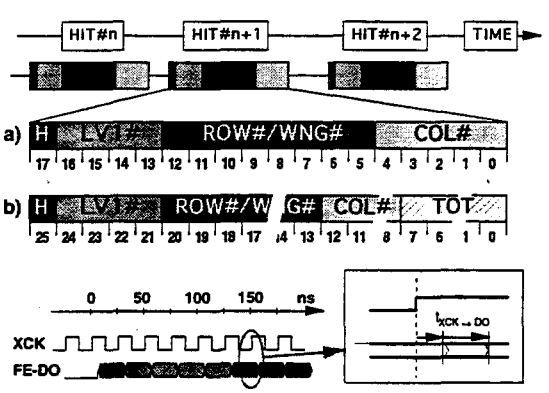

Figure 5: Hit Data format at the input of the MCC. Case b) illustrates the Time over Threshold encoding.

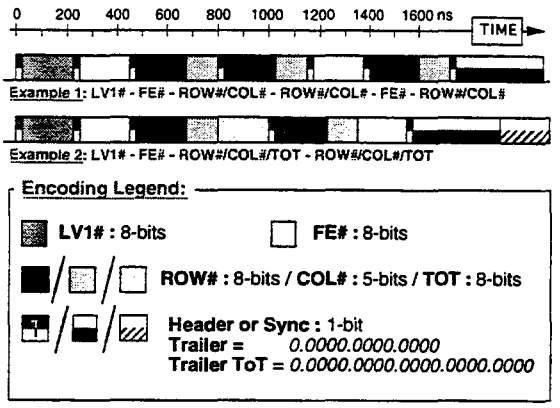

Figure 6: Event Data format at the MCC output. Example 2 illustrates the case of Time over Threshold encoding.

difference can be programmed in the ROD to take into account timing differences. The XCK signal is used by pixel Front End to latch and associate track hits to a particular bunch crossing number.

- LV1 (Level 1 Trigger)

  This signal validates the rising clock edge of the bunch crossing clock (*FE-XCK*) for those crossings that have been accepted by the level 1 trigger system. The serial control protocol is used to transmit the LV1 command from the counting room down to the MCC, where it is received serially encoded on the *MCC-DCCI* pin. Once received by the MCC this signal is decoded by the *Command Decoder* inside the MCC and distributed to all the FE chips. One of the features of the MCC is the ability to generate a pattern of contiguous LV1 Accept signals to allow possible multi-crossing read-out, both for diagnostic and timing initialization, and to allow for the possibility that all of the interesting hits may not be available in a single beam crossing. Hits belonging to triggered bunch crossings cause data to be stored by the End-of-Column logic in the FE chips and subsequently pushed to the MCC. The MCC puts together hits from the same event and pushes them to the ROD's.

- SYNC (Event Synchronization)

  This signal is used to automatically re-synchronize all FE chips in a module in case of error conditions. The MCC can generate this signal autonomously whenever the particular module is empty, or when it detects an error condition. It can also receive a SYNC command from the ROD. In the first case all the FE chips on the module controlled by the MCC which issued the SYNC reset their internal LV1 counter and delete any pending triggered event. The FE chips will then resume data taking and pushing as soon as they see the next LV1 signal. In a similar way, all MCC's respond to a SYNC sent by the ROD by resetting all FE chips connected to them and resetting the LV1 counter and data inside the FIFO's once as many pending events as possible have been reconstructed. The MCC then sets a warning condition in this events data streams.

- Slow Commands

  These commands are used to program the MCC or the FE chips in a particular module. When one of these commands is issued, the pixel system is taken out of data acquisition mode, and command operations like system configuration or read/write operations can be executed at any level of the hierarchy. There is a Data-Take command in order to resume data taking.

4. Data Formats

Data will be transmitted from the FE chip to the MCC using a simple protocol in which each hit is transmitted in a stream of 18 bits (26 if the optional Time-over-Threshold (ToT) information is produced): 1 header bit, 4 LV1 bits, 8 row number or end-of-event/warning bits, 5 column number bits and 8 optional ToT bits. Individual hits can be separated by any number of zeroes in the data stream. The average occupancy of the link is expected to be fairly low. This, together with the encoding scheme

described here, permits automatic recovery after data transmission errors, so long as a gap longer than 18 (or 26 in case the ToT option is selected) bits appears in the data stream. The data encoding and transmission is represented in Figure 5.

The format of the event generated by the MCC is made of ordered fields separated by synchronization bits ("1"), starting with a header ("1") and ending with a trailer ("100 0000 0000 0000"). The header is to "wake up" the receiver from the idle condition when no data are transmitted, while the synchronization bits together with the trailer are used to define the end of event. The trailer has been chosen to be a pattern that can never occur in the data stream, due to the insertion of synchronization bits. The format always requires a LV1 number at the beginning followed by a sequence of FE chip numbers, followed by any number of hits (row plus column or row plus column plus ToT when this option is selected) interrupted by a new FE chip number plus the sequence of hits. Warning and error flags can be introduced in the event format. An example of such a format is given in Figure 6.

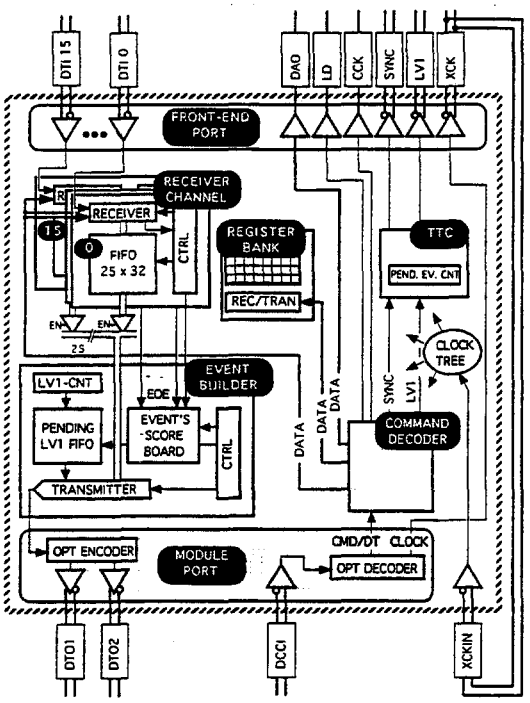

Figure 7: MCC Block Diagram.

5. MCC Architecture Implementation

The MCC architecture is shown in the block diagram of Figure 7. The blocks represented are:

FRONT END PORT: (I/O to the FE chips)
This module performs all of the interface functions between the FE chips and the MCC.

RECEIVER CHANNEL: (Front End receiver)
This block includes the 16 derandomizing FIFO's (one for each FE chip output), made with a full custom memory of 32 25-bit long words with arbitration logic and read/write pointers, and the FE receivers and the state machine running the interface to the FE link. FIFO overflows are handled by this block.

EVENT BUILDER:
A scoreboard keeps track of complete events read by each FE chip. Once all 16 FE chips have sent a complete (even empty) event, the *Event Builder* sorts, formats and prepares the event stream to be transmitted to the RODs once encoded by the *Module Port*. Errors at the level of event building are dealt with by this block. Both `error` and `warning` conditions are added in the output data stream if necessary.

REGISTER BANK:
The MCC architecture provides up to 16 general purpose registers, only 11 of which have been implemented in the demonstrator MCC. As an example, a register is used to mask one or more input lines coming from FE chips in case of failure. Two special registers contain the bit length of commands and data to be written to the Front End chips as these quantities may vary each time a FE chip is accessed.

COMMAND DECODER:
This block decodes all commands sent to the MCC. The LV1 trigger command (fast command), as well as read/write operations to internal MCC registers or to FE chips (slow commands) are amongst the commands interpreted by this block.

TTC: (Trigger, Timing and Control)
This block generates the LV1 trigger to the FE chips. It has the ability to block LV1 accepts to all of the FE chips on a module if there are too many LV1 events still to be transmitted to the MCC. This block therefore keeps track of all LV1 signals sent to the FE chips and all the Event-Done signals received by the Event Builder. One of the functions of this block is to deal with the automatic Sync signal programmable into the MCC or received by the RODs via a dedicated command.

MODULE PORT: (Interface to the ROD's)
This module performs all of the interface functions between the MCC and the RODs. Clock and serial commands are decoded and sent respectively to the *Clock Tree* and the *Command Decoder* block. Outgoing signals are encoded with the clock and are sent out on *MCC-DTO1* and *MCC-DTO2* pins according to bandwidth requirements. Bandwidth requirements predicate the use of three optical fibres.

CLOCK TREE:
Once the clock has been decoded by the *Module Port*, it must be distributed to the whole module. In order to minimize clock skew between the MCC and the FE chips, the MCC generates the clock and latches all input/output data with this clock signal. This signal is then sent to all the components of the module via the *MCC-XCK* output pin. The MCC itself will use this distributed clock for all its internal operations.

The MCC chip will be fabricated in rad-hard technology using the DMILL 0.8 μm double metal CMOS process. To minimize the risk of errors in the rather complex MCC design, a top-down design methodology has been used, using standard cells with logic synthesis from a high level Verilog description. The only full custom parts of the design will be the 16 FIFO's, the LVDS I/O drivers/receivers and the Bi-phase Mark Encoding and Decoding blocks, which do not exist in the DMILL standard cell library.

6. MCC for the ATLAS "Demonstrator"

Since the demonstrator module does not have optical links, the main differences between it and the final version are the absence of the Bi-phase Mark Encoding and Decoding blocks which will be added to the Module Port (see Figure 8). Another difference is that this version of the chip has been produced in rad-soft technology using the AMS 0.8 μm double metal CMOS process.

Due to the complexity of the circuit a top-down design methodology has been chosen. The whole design was first coded in Verilog at a behavioural level, and then mapped to the 3.3 V, 0.8 μm CMOS AMS standard cell library using a logic synthesis tool (Synergy), with the exception of the full-custom blocks. After hand-positioning the custom blocks, the layout was made with an automated place and route tool. The final steps of this process were the layout versus schematic comparison and the design rule check. The whole design was made using the Cadence Design Framework. Figure 11 presents the final

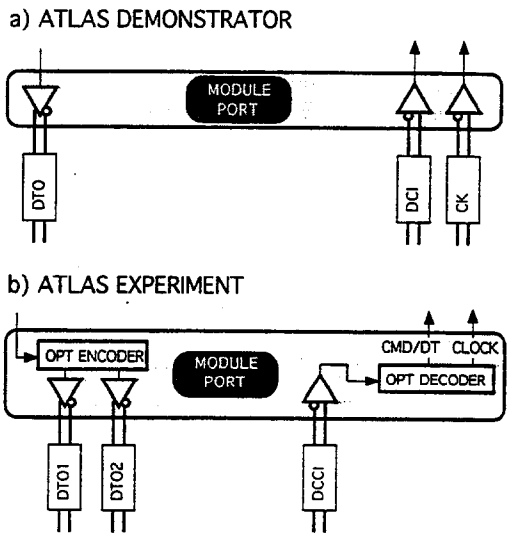

Figure 8: Difference between the "Demonstrator" MCC (a) and the MCC which will be designed for the experiment (b), The final chip will receive clock and data encoded on a single fiber (DCCI), while two output fibers will be used to increase the bandwidth (DTO1 & DTO2).

layout with the main building blocks highlighted. Approximately half of each receiver channel foot print is occupied by a FIFO.

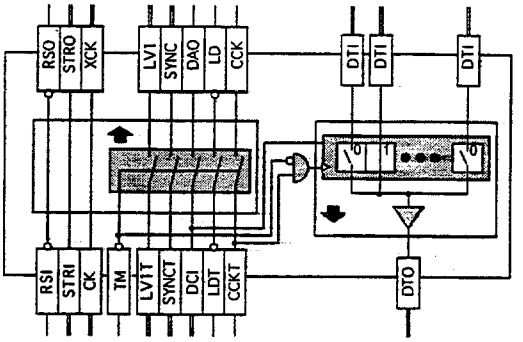

Figure 9: Block diagram to illustrate the "Transparent Mode" operation of the MCC.

Since the 16 FE chips in a module will be accessed through the MCC by a rather complex protocol, we have added a "Transparent Mode" of operation to the MCC. This mode requires the addition of several pins and a small amount of logic, but allows transparent access to all input and output pins of the 16 FE chips. This operational mode bypasses all of the MCC

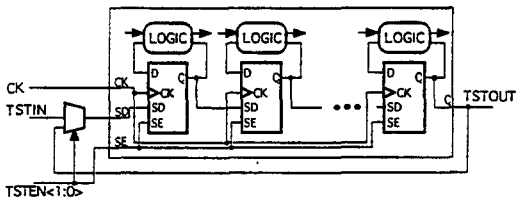

Figure 10: Scan chain to access in read and write mode MCC internal flip-flops.

functionality and directly connects all data, control and timing signals going to (or returning from) the FE chips with corresponding externally accessible pins on the MCC.The Figure 9 shows the implementation of the "transparent Mode" in the MCC.

The MCC has additional pins which can be used to put the circuit into test mode and increase the accessibility and observablity of internal nodes. In this test mode all the internal flip-flops are configured as a shift register, see Figure 10, which allows both reading and writing the MCC internal state. This mode will be used to test the chip on a probe station before mounting it in the system.

The MCC has 60 signal pins, and with the inclusion of power and test signals, there will be a total of 81 pins on the chip die. The chip area is 67 mm^2 and has approximately 430,000 transistors. The design will not be optimal in its area and power consumption, but this approach provides the safest route to produce the MCC prototype on this short time scale.

The MCC is in fabrication in an engineering run, and we will receive functionally-tested chips, in order to facilitate the assembly steps of the whole module. A first test of the whole module should take place in August 1998 in the CERN H8 test beam.

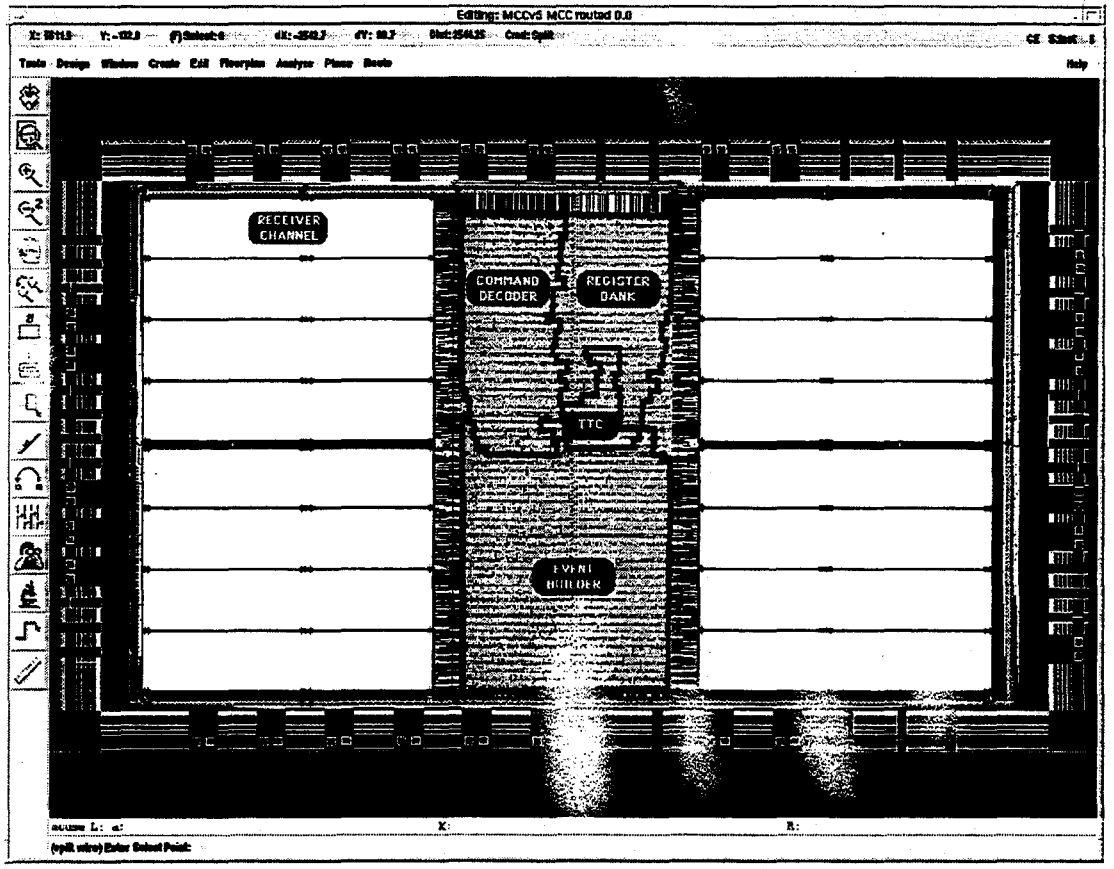

Figure 11: MCC layout. The different functional blocks have been highlighted.

Aknowledgments

I wish to thank the co-authors of the MCC demonstrator chip design: R. Beccherle, G. Comes, G.Gagliardi, G. Meddeler and P. Musico.

References

[1] ATLAS Pixel Collaboration: M. Ackers, M. S. Alam, M. Aleppo, E. Anderssen, A. Andreazza, B. Athar, D. Barberis, R. Beccherle, C. Becker, K.-H. Becks, D. Bintinger, L. Blanquart, W. Boyd, A. Brandl, G. Cabras, M. Caccia, D. Calvet, C. Caso, D. Cauz, E. Charles, S. Ciocio, J-C. Clemens, D. Cobai, G. Comes, M. Dameri, J. Dailing, G. Darbo, S. D'Auria, A. De Angelis, B. De Lotto, C. del Papa, P. Delpierre, D. Dorfan, J. Drees, T. Dubbs, K. Einsweiler, J. Emes, T. Fahland, D. Fasching, P. Fischer, G. Gagliardi, Y. Gally, C. Geich-Gimbel, C. Gemme, P. Gerlach, M. Gilchriese, K.W. Glitza, M.S. Gold, S. Gonzalez, G. Gorfine, C. Gößling, I. Gregor, A. Grillo, P.Gutierrez, O. Hayes, G. Hallewell, M. Hoeferkamp, M. Holder, F. Hügging, R. Jared, A. Joshi, M.Keil, S. Kersten, S. Kleinfelder, R. Kluit, M. Kobel, Z. Kohout, T. Kuhl, A.J. Lankford, R. Leitner, G. Lenzen, C. Linder, Z. Ling, A.K. Mahmood, R. Marchesini, J.A.J. Matthews, F. McCormack, T. McMahon, G. Meddeler, C. Meroni, S. Meuser, O. Milgrome, P. Morettini, T. Mouthuy, P. Musico, J. Nachtman, K. Neiyanuri, W. Ockenfels, M. Olcese, B. Osculati, N. Palaio, Y. Pan, F. Parodi, F. Pengg, S. Pier, E. Piotto, P. Polevin, R. Potheau, A. Pozzo, F. Ragusa, J. Richardson, G. Ridolfi, T. Rohe, L. Rossi, A. Rozanov, H.F.W. Sadrozinski, L. Santi, R. Schoefer, I. Scott, F. Scuri, S.C. Seidel, A. Seiden, G. Sette, H.S. Severini, P. Skubic, P. Sicho, P.K. Sinervo, J. Snow, B. Sopko, E. Spencer, D. Stoker, M. Strauss, L. Stupka, J. Thadome, S.C. Timm, L. Tomasek, J. Treis, W. Trischuk, C. Troncon, V. Vacek, I. Valin, B. van Eijk, G. Vegni, E. Vigeolas, V. Vrba, F. Waldner, F.R. Wappler, N. Wermes, S.L. Wu, R. Wunstorf, J. Wüstenfeld, M. Ziolkowski, G. Zizka, H. Zobernig.

[2] ATLAS Pixel Detector Technical Design Report, CERN/LHCC/98-13.

[3] G. Gagliardi, The ATLAS Pixel Detector, these proceedings.

[4] G. Darbo, The ATLAS Pixel Detector System Architecture, Proceedings of the "Third Workshop on Electronics for LHC Experiments", CERN/LHCC/97-60.

[5] G. Darbo et al., "ATLAS Pixel Demonstrator: Pixel Electronics Specification", ATLAS INDET Note in preparation, http://www.ge.infn.it/ATLAS/Electronics/home.html.