Fermi National Accelerator Laboratory

FERMILAB-Conf-98/278

High Readout Speed Pixel Chip Development at Fermilab

G. Cancelo, D. Christian, J. Hoff, A. Mekkaoui, R. Yarema and S. Zimmermann

Fermi National Accelerator Laboratory
P.O. Box 500, Batavia, Illinois 60510

October 1998

Published Proceedings of the *Fourth Workshop on Electronics for LHC Experiments*,
Rome, Italy, September 21-27, 1998

Operated by Universities Research Association Inc. under Contract No. DE-AC02-76CH03000 with the United States Department of Energy

Disclaimer

Distribution

Copyright Notification

HIGH READOUT SPEED PIXEL CHIP DEVELOPMENT AT FERMILAB

G. Cancelo, D. Christian, J. Hoff, A. Mekkaoui, R. Yarema, S. Zimmermann
Fermi National Accelerator Laboratory, Batavia, IL 60510

Abstract

Pixel detectors are becoming a very important part of high energy physics experiments, including those at the Tevatron and the LHC. At Fermilab, a pixel detector for the BTeV experiment is proposed for installation a few millimetres from the beam. Its information will be used in on-line track finding for the lowest level trigger system. This application requires pixel chips with high readout speed. The architecture of the pixel chips being designed at Fermilab will be presented, and future proposed developments and simulations will be summarised.

1. INTRODUCTION

At Fermilab, the BTeV experiment has been proposed for the C-Zero interaction region of the Tevatron [1, 2]. The innermost detector for this experiment will be a pixel detector composed of 93 pixel planes of 100×100 mm each, divided in 31 triple-stations perpendicular to the colliding beam and installed a few millimetres from the beam. This detector will be employed for on-line track finding for the lowest level trigger system [3] and, therefore, the pixel chips will have to read out all detected hits. Simulations have shown that, given a luminosity of 2 $\times 10^{32}$ $cm^{-2}s^{-1}$ (which corresponds to two interactions per crossing), a pixel chip of 8×7.2 mm active area placed 6 mm from the beam (the innermost chip), will be hit by one or more tracks in approximately one 132 ns bunch crossing out of four [4]. At this luminosity, it is estimated that an average of approximately five 50×400 μm pixels will be hit in the innermost chip in those crossings with any data [5]. Therefore, this pixel chip has to sustain an average readout rate of 1.25 pixels per BCO. To account for statistical fluctuations and other effects (for example, the same simulations have shown that more than 20 pixels can be hit in just one BCO) the chip has to be capable of even higher data transfer performance.

Another very important factor that impacts the required data transfer rate is the need for analog to digital conversion (ADC) of the detected pulse height. Simulations have shown that a 50×400 μm pixel with a two or three bit ADC may be enough to achieve the necessary resolution [6]. Experiments that are now being arranged for a test beam should help to confirm the final chip requirements. Though there is no final agreement about the pixel size within the BTeV collaboration, there is a reasonable consensus that the experiment will require analog readout.

The pixel chip will be installed very close to the beam and therefore will have to be implemented in a radiation hard technology. We intend to use the new 0.5 μm process from Honeywell to accommodate this very severe constraint. Prototype chips will be made using the Hewlett Packard 0.5 μm CMOS process.

The pixel chip development described here is a succession of steps and submissions toward a chip that meets the BTeV requirements, each achieving specific engineering goals. The chips resulting from these steps have been dubbed FPIX0, FPIX1, and so on. In the next sections we will describe FPIX1, which represents the first step towards the final pixel readout architecture necessary for the BTeV experiment. Its primary purpose is to determine how fast the chip can process information internally and therefore, allow accurate extrapolation to the ultimate possible readout speed. Previous steps included FPIX0 [7, 8] and Pre-FPIX1, which were designed to test different front-end configurations and cross-talk management ideas. We anticipate that enhancements to the FPIX1 architecture will be necessary to achieve the readout speed required by BTeV. Simulations based on our experience with FPIX1 will guide the design of these enhancements.

2. FERMILAB PIXEL CHIP 1 (FPIX1)

The FPIX1 is a column based pixel chip with 50×400 μm pixel cells arranged in an array of 160 rows by 18 columns. Similar to other pixel chips [9], it uses an indirect addressing scheme to reference pixel hits to beam crossing (BCO) numbers. However, unlike others, it does not use pointers to accomplish the indirect addressing. Instead, it utilizes a Command Driven Architecture sufficiently unique to warrant description. The chip can be divided into three mutually dependent pieces: the Pixel Cell, the End-Of-Column (EOC) Logic and the Chip Logic [10] (Figure 1). The responsibility of the Chip Logic is to control and maintain all features that are common to the chip such as the clocks, the "current" and "requested" BCO number, and the status of off-chip communication. Each one of the eighteen EOC Logic cells controls one column. They do so by responding to information from the Chip Logic and from the 160 pixels each one control, by broadcasting commands to the Pixel Cells and by arbitrating with the other EOC Logic cells for control of the on-chip buses. Finally, each Pixel Cell connects to one pixel detector and respond to commands

Work supported by the U.S. Department of Energy under contract No.DE-AC02-76CH03000. Fermilab conference paper No. 98/278.

from the EOC Logic. The commands used in this architecture are the following. The "input" command instructs a Pixel Cell to accept hits from its pixel detector and to respond to such a hit by alerting the EOC Logic to its arrival. In the absence of and input command a hit is ignored. The "output" command instructs the Pixel Cell to prepare to write its information onto the bus. The "reset" command instructs the Pixel Cell to reset its contents. Finally, the "idle" command instructs the Pixel Cell to do nothing.

Each EOC Logic consists of four EOC Sets each one capable of generating its own commands. When a Pixel Cell receives a hit, it immediately associates itself with whatever EOC Set is broadcasting the "input" command. From that point until it is reset or output, the Pixel Cell only responds to commands from its associated Set. Meanwhile, the EOC Set saves the timestamp. The EOC Set can then issue the "output" command (to readout the hit information) or "reset" command (to reset the hit inside the Pixel Cell). Observe that the information stays stored inside the Pixel Cell and the EOC Set until readout.

The Chip Logic supports two readout modes. The first one, the "continuous" readout mode, is done automatically by the FPIX1 chip without the need of an external trigger. This mode is planned for the BTeV experiment. The other is the external trigger mode, in which an external system must provide the timestamp of the hits that should be readout. This mode is applicable for pixel detectors with external trigger or for diagnostic purpose. Throughout this paper, we will refer to the continuous mode unless we specifically note otherwise.

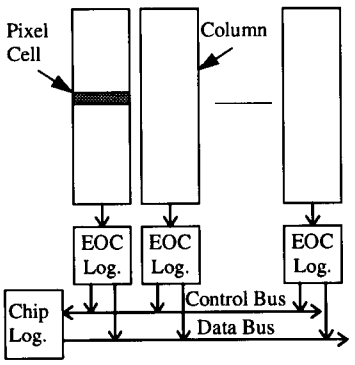

Figure 1. FPIX1 Block Diagram

2.1 Pixel Cells

The pixel cells hold the front-end electronics and the digital interface with the EOC Logic. The final proposed configuration for the front-end was done based in FPIX0 and Pre-FPIX1 Hewlett Packard CMOS process submissions where different aspects of the front-end electronics, like noise, crosstalk, threshold dispersion, etc. were tested. Reference [8] reports in details the results of the tests done with FPIX0. The front-end (Figure 2) contains a charge sensitive amplifier (CSA) and a second amplification stage. The output of the second stage connects to a flash ADC and discriminator. The DC feedback used in the CSA is similar to the one described in [11]. The average discharge time of the CSA can be externally set from 50 ns to 1 ms by an external current source using the feedback current [8], without requiring any reset signals transmitted across the sensitive analog region.

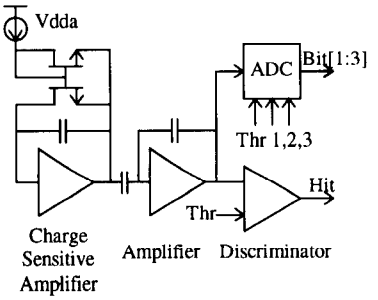

Figure 2. Front-end

The discriminator output Hit is asserted when the signal in the input of the CSA is higher than threshold (Thr). The flash ADC is formed by three comparators which directly connect to dedicated SR flip flops (FF) inside the Pixel Cell. The thresholds for these comparators are externally provided by inputs Thr1, 2 and 3. During readout, tristate buffers connect the outputs of the FFs (Bit[1:3]) to the EOC Logic where they are encoded into two bits, forming a binary encoded two bit ADC pulse height: ADC[1:0]=0 corresponds to Bit[3:1]=0, ADC[1:0]=1 to Bit[3:1]=1, ADC[1:0]=2 to Bit[3:1]=3 and ADC[1:0]=3 to Bit[3:1]=7. Therefore, ADC conversions from 1 to 3 corresponds to values above threshold.

The digital interface of the pixel cell is depicted in Figure 3. It has two major components: the Command Interpreter and the Pixel Token and Bus Controller. The Interpreter has inputs for four EOC Command Sets, each corresponding to one of the four internal EOC Sets. As we already stated, the types of EOC Commands are input, output, reset and idle and they are delivered simultaneously to all Interpreters of the same column. When the Interpreter is executing the input command and the Hit output from the discriminator is asserted, the Interpreter alerts the EOC logic, via the HFastOR signal, that some cell in the column was hit. The HFastOR line is the wired-OR of all HFastOR outputs of the Command Interpreter. This operation is executed independent of the Master Clock (Mclk). However, from now on the readout proceeds synchronous with the Mclk. At the raising edge of Mclk, the EOC Logic delivers the output command and the Interpreter belonging to the pixel cell previously hit requests the bus via the bus request signal (Breq). The Interpreter also asserts the RFastOR line, which, like the HFastOR alerts the EOC Logic that some cell in the column still needs to be read out. EOC Logic provides a column token on the bottom of the column as a means to regulate bus access. The column token ignores pixel cells with no information until it reaches a cell that is requesting the bus. This propagation to an interesting

pixel is done in less than one clock cycle, even if the pixel that was hit is the last in the chain. At the next rising edge of the MClk, the pixel with interesting data and the column token will load its data onto the bus and drive it to the EOC logic for one clock cycle. In parallel, the column token is transmitted to the next interesting cell, pipelining the output of the Pixel Cell with the token passing. This allows the readout of one Pixel Cell per clock cycle, without any wasted MClk cycles. The data is composed of the ADC count Bits[3:1] and the row address Radd[7:0]. As the interesting cell is readout, it automatically resets itself and withdraws its assertion of the RFastOR. The RFastOR will return to its inactive state when all of the interesting pixels have been read. This way, the EOC Logic is able to detect when the last interesting pixel in the column is being output. At the next rising edge of the MClk, control of the on-chip bus is transferred to the next interesting column.

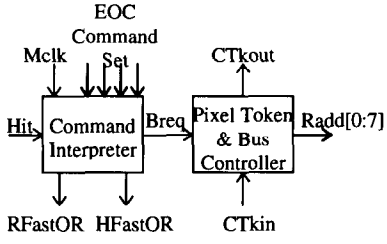

Figure 3. Pixel Cell Digital Interface

At any given time, only one EOC Command Sets is permitted to broadcast the input. By this mechanism, the pixel cell dynamically associates itself with only one EOC Set. After the association has been made, that specific cell ignores the commands from all other Command Sets, and now just monitors commands issued by that specific EOC Set. Therefore, no other input operation can be executed by the cell, and the pixel hit information stays stored inside the cell until its readout. Other pixel cells that were not hit will continue to monitor the Commands Sets, waiting for coincidence of hit and input commands, and the column will remain active, while the readout on the pixel cells previously hit can proceed.

2.2 *End of Column Logic*

Figure 4 shows a block diagram of the EOC Logic. It consists of a Priority Encoder and four EOC Sets. The EOC Sets themselves consist of one EOC Timestamp register, an EOC state machine for generating the appropriate EOC commands and two comparators.

The priority encoder chooses which of the EOC Set will issue the input command. When there is a hit somewhere in the column, the HFastOR signal is asserted, and the state machine inside the assigned EOC Set responds by latching the Current BC0 (CBC0) inside EOC Timestamp register and by issuing the idle command at the next rising edge of the BC0 clock. This ensures that all pixels in a particular column hit in the same clock period are associate with the same End-of-column Set. This "hit" EOC Set now waits for matches to broadcast commands. If the match is between the Request Bco (RBco) and its latched CBco, the EOC Set broadcasts the output command, and if the match is between the CBco and its latched CBco, it broadcasts the reset command. Meanwhile, the priority encoder assigns the next EOC Set to issue the input command to the column. Since the EOC Logic has four EOC Sets, the pixel cells on the column can record up to four different timestamps before it runs out of EOC Sets. Observe that the broadcast of the reset command is useful for the external trigger mode, where hits that were not readout by some externally provided timestamp need to be reseted.

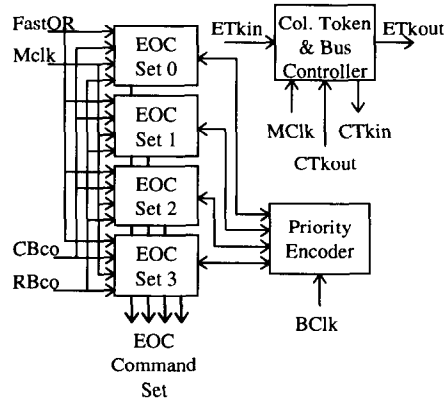

Figure 4. End Of Column Logic

A second state machine is implemented inside the Column Token and Bus Controller, to control the access to the EOC data bus. This access is arbitrated by an EOC token. As soon as there is a match between the RBco and the latched CBco, the Column Controller issues the CTkin token to the column, and waits for the EOC Token In (ETkin) from the Chip Logic. When the ETkin is asserted, the Column Controller enables the pixel data into the internal data bus, and stays in this state until all hits in the column are readout. The Column Controller now passes the EOC token to the next EOC Logic by asserting ETkout. The early delivered of the CTkin to the column, even before the ETkin was received, allows for the pixel data to be asserted in the internal data bus as soon as the EOC token arrives to the EOC Logic. This, added with the assertion of ETkout as soon as CTkout is received by the Column controller allows for full clock speed readout of the pixel data, now even when the chip finishes the readout of one column and starts the readout of the next. Finally, the operations associated with readout are synchronous with MClk, while operations associated with the pixel cell hit are synchronous with BClk or with the hit itself.

2.3 *Chip Logic*

The Chip Logic controls the features associated with the whole chip. It is basically formed by the Current BCO counter (CBco) and the Readout BCO counter, a multiplexer and a chip controller. The CBco increments

synchronously with the BClk and is delivered to the EOC logic. The multiplexer multiplexes the Readout BCO counter or the External Request BCO (in case external trigger is used). The output of the multiplexer forms the RBco number that is delivered to the EOC logic. When the chip is operating in the "continuous" readout mode, the multiplexer connects the Readout BCO counter to the RBco, and this counter will provide the timestamp number that should be used to compare with the timestamp latched inside EOC Timestamp register. The clock of the Readout BCO counter is not a free running clock. The Readout BCO counter always is two counts behind the CBco, in order to avoid comparisons in the EOC logic of some event that did not yet stabilize inside the pixel cell. It counts in parallel to the CBco counter (using the BClk) until the EOC Sets detect a match between the RBco and the timestamp latched inside the EOC Timestamp register. Then the clock stops and the chip controller starts the readout of the chip by issuing a ETkin to the first EOC Logic. When the readout of that specific timestamp finishes (i.e., the Chip Logic detects that ETkout of the last column was asserted), the controller quickly increments the Readout BCO Counter (using the MClk) until another match is detected or the Readout BCO counter reaches two counts behind the CBco. This feature allows for the data to be read aligned by timestamp.

Other features of the Chip Logic includes the bus arbitration, which is done again by a token passing from chip to chip, the control of the configuration of the chip and data "throttling". The configuration is accomplished by a serial bit stream that programs features like pixel cell kill (to disable noisy pixels), pulse inject select (for enabling programmable pixel cells to accept charge inject directly into the front-end using an external voltage source), and other programming features. Data throttling is the following: there is an external input to the pixel chip that allows to command the pixel cell to disregard some bunch crossing, even if it is hit. Therefore, when the DAQ is reading the chip, it can monitor the timestamp of the data. If the timestamp is getting too delayed in comparison to the current BCO, creating the possibility of recorded hit losses, the DAQ can command the pixel chip to disregard some random future bunch crossings (to avoid data bias), until the delay drops to some suitable value.

3. FERMILAB PIXEL CHIP 2 (FPIX2)

Before we can choose the final FPIX2 architecture, we need to decide how many bits of analog information is actually necessary for the BTeV experiment. The present plan is to use FPIX0 in a test beam, bump bonded to Atlas detectors, to collect information that should give experimental basis for this decision. As we already described, FPIX1 implements a two bit flash ADC inside the pixel cell, and we have confidence that a three bit ADC could fit inside a slightly larger cell, as for example 40×450 μm. However, if the test beam shows that four or more bits of analog to digital conversion is required, the ADCs will have to be moved to the periphery of the chip, and the analog pulse height information will have to be transmitted for digitization along the columns.

However, the uncertainty of the number of bits of the ADC does not preclude us to propose and simulate features to include in FPIX2 to increase the readout speed. We will now present some of them. Clearly, one possibility is to increase the readout clock frequency (MClk for FPIX1), but this is constrained by the internal speed of the chip and by power dissipation. Another option is to increase the width of the output data word, but this is also constrained by the complexity and mass of the multichip module interconnect and the chip power dissipation. A third option is to achieve some form of data compression inside the chip, i.e., the chip still transmits the same amount of information, but with fewer data cycles. The data alignment by timestamp implemented in FPIX1 already achieves an initial degree of data compression. During readout, the timestamp has to be transmitted just once for all the hits that occurred simultaneously. We are considering to implement in FPIX2 what we have named "group" reading. The concept is the following: instead of reading a column by individually reading each pixel cell, FPIX2 will read groups of consecutive pixel cells in parallel. So, in this sense, we will not have pixel row address, but actually pixel group address, and the row position of several pixels can be uniquely identified with just one row address. Simulations described in next section have shown a data compression by a factor of 2.36 with respect to a pixel by pixel readout. Other options for data compression includes the transmission of the column address just once for all the data available inside a column, again, requiring the data transmission of just one column address for several groups in the same column.

Another method to increase the readout speed is to use a higher readout clock rate. In this case, all the chip operates at some lower frequency attainable with the microelectronics process and some reasonable power dissipation, and just the portion associated with the output data bus operates at some higher rate (for example, the readout clock operates at twice the frequency of the internal clock). Another problem then arises: how to deliver enough hit information to the output data logic in order to optimize the utilization of the data bus, i.e, if the chip has hits and has control over the data bus, it transmits data continuously over the bus without wasting any readout clock cycle. We have considered two options toward this goal. One is to use multiple buses inside the pixel chip, to transfer the hit information of multiple columns in parallel to the output data logic. The output data logic then multiplexes at higher rate the hit data of different internal buses to the output data bus. This approach does not look attractive to the BTeV experiment, since simulations have shown that, in 60% of the BCOs

with tracks, just an individual column is hit, and simultaneous readout of multiple columns will in general not avoid waste of readout clock cycles [5]. The second option is to use a very wide internal data bus inside the chip, i.e., the chip transmits all information associated with a group (group column and row address and four ADC conversions) in one internal clock cycle and in parallel to the data output logic. The output logic then divides this word into narrower words, and transmit them at higher frequency rate. This second approach looks very attractive, since it addresses the concern previously raised by the simulations. Furthermore, it takes advantages of facilities already implemented inside FPIX1, and in specific, the set of pipeline features that allows the chip to read at full clock speed, without wasting any clock cycle, even when the chip finishes the readout of one column and starts the readout of the next.

Finally, we are also considering the pipeline of the Readout BCO Counter with the readout of previous timestamp, or even some faster method to locate the next stored timestamp. Other issues associated to error flags will also be included in the chip.

4. SIMULATIONS

We have done extensive simulations of FPIX2 proposals assuming different chip architectures. We will describe here the simulations that assumes two or three bit flash ADCs inside the pixel cells. For extended results see Reference [12]. The objective of the simulations were to estimate if the proposed chip architecture can achieve the necessary data rate to avoid hit loss. The conditions for the simulations were as follows. The delays used for the internal logic of the chip are compatible with Verilog and Spice simulations of FPIX1. We simulated the performance of the pixel chip which will receive the maximum fluence in the detector. We assumed a pixel size of 50×400μm and a chip with 160 rows by 18 columns, four pixels "grouping", a wide internal bus operating at 26.5 MHz and an output bus operating at 53 MHz with a half the width of the internal bus. As hit inputs we used Monte Carlo simulations of minimum bias and b-quarks events of approximately 5000 BCOs, collected charge of electrons and chip threshold set at 2000 e^-. The total number of interactions per BCO is a random number chosen from a Poisson distribution with mean 2. This is equivalent to a luminosity of 2×10^{32} $cm^{-2}s^{-1}$, the highest luminosity expected for the BTeV experiment. The chip is inside a 1.6 T magnetic field [5]. We had a total of 1240 BCOs with hits, and a total of 6152 hits, which represents an average of approximately five pixel hits per bunch crossings with tracks. The results of the simulations show that reading groups of four consecutive pixels altogether make a data compression improvement of 2.36 with respect to a pixel by pixel readout. Also, FPIX2 can easily handle the data rate, with approximately 40% utilization of the output data bus. This give us a reasonable margin to account for other effects not included in the Monte Carlo simulations.

5. CONCLUSION

In this paper we have described the FPIX1 chip and design issues to achieve full internal readout speed. Then we described proposed enhancements which we plan to incorporate in FPIX2 in order to increase data compression and readout speed. Finally, simulations have demonstrated that the FPIX2 architecture should achieve the hit readout rate required by the BTeV experiment.

6. REFERENCES

[1] Kaplan, D. M., "BTeV/CO," Proc. Int. Symp. on Near Beam Physics, pp. 66-71, Fermilab, 1998.

[2] Santoro, A., *et.al.*, "An Expression of Interest for a Heavy Quark Program at C0," BTeV pub. note, Fermilab, May 1997.

[3] Husby, D., *et.al.*, "Design of a secondary-vertex trigger system for a hadron collider," Nucl. Instrum. Meth. A383, pp. 193-198, 1996.

[4] Kasper, P., "Study of the pixel occupancy in the BTeV detector," BTeV int. note, Apr. 98.

[5] Kasper, P., Pixel hit data generated from Monte Carlo simulations for minimum bias and b-quarks events, Fermilab, Mar. 1998.

[6] Artuso, M. and Wang, J., "Some further studies on factors affecting the pixel resolution," BTeV int. note, July 1998.

[7] Mekkaoui, M., *et.al.*, "Results from the FPIX0 chip bump-bonded to the ATLAS pixel detector," to be presented in the same workshop.

[8] Mekkaoui, A., "FPIX0: A prototype pixel FE chip at Fermilab," Proc. 3rd Int. Meeting Front End Elect. for High Resol. Tracking Det., Taos, NM, Nov. 1997.

[9] Wright, M., Millaud, J., and Nygren, D., "A pixel unit-cell targeting 16ns resolution and radiation hardness in a column read-out particle vertex detector," LBL-32912, Berkeley, 1992.

[10] Hoff, J., "FPIX1 architecture and operation: design and simulations," PPD/ETT/ES int. note, Fermilab, June 1998.

[11] Blamquart, *et.al.*, "Pixel analog cells prototypes for ATLAS in DMILL technology," Nucl. Instrum. Meth. A395, pp. 313-317, 1997.

[12] Cancelo, G., Zimmermann, S., "Modeling and simulation of a readout architecture for pixel-based detectors," to be presented in the 1998 Nuclear Science Symp., Toronto, Canada, Nov. 1998.