The AM++ Board for the Silicon Vertex Tracker Upgrade at CDF

A. Annovi, A. Bardi, M. Bitossi, R. Carosi, M. Dell'Orso, P. Giannetti, P. Giovacchini, I. Pedron, M. Piendibene, L. Sartori, F. Schifano, B. Simoni, F. Spinella, S. Torre, and R. Tripiccione

***Abstract*—A critical element of the Silicon Vertex Tracker upgrade is a new Associative Memory (AM) $\sim$40 times larger than the existing one based on application-specific chips. The Associative Memory is the SVT module performing pattern recognition. We have developed a new Associative Memory board (AM++) to handle 128 application-specific chips. We describe the design, the tests, and the performances of the new AM++ for CDF.**

***Index Terms*—Parallel processing, particle tracking, pattern matching, triggering, very large scale integration.**

I. Introduction

DEDICATED hardware systems, based on the use of an Associative Memory (AM) [1], [2], have been proposed for track recognition in high-energy physics experiments.

A programmable AM based on a modular architecture has already been realized using full custom ICs [2] and Field Programmable Gate Arrays (FPGAs) [3].

In this paper, we describe a new Associative Memory board (AM++) using application-specific standard-cell associative memory chips (the Amchip03 device also called the AMchip) [4] that we have developed for the Silicon Vertex Trigger (SVT) [5] upgrade at the Collider Detector at Fermilab (CDF) experiment. The new AM system increases the SVT computational power for high luminosity runs.

The AM must be able to store all trajectories of interest and extract the ones compatible with a given event. A trajectory is compatible with an event if all (or a majority of) detector channels crossed by that trajectory have fired in that event. In order to contain the trajectory memory within an acceptable size, the AM operates at a coarser resolution than the actual detector. This is usually done by clustering single contiguous detector channels into larger superstrips. In the following, we will call "hits" the addresses of fired superstrips in each layer, and "roads" the coarse resolution trajectories (each road corresponds to an array of superstrip addresses—one per detector layer). The AM finds roads linking a drift chamber track (XFT layer) and the requested number of silicon hits. Five silicon layers are used. Hit combinations inside roads are fitted to find real tracks measured with the best detector resolution.

SVT processing time increases as the instantaneous luminosity rises. Multiple interactions in the same accelerator bunch crossing increase the number of hits in the SVX, and thus increase the time to process all of the silicon hits (time/hit $= \sim$35 ns). An additional, more serious problem is an increase in the number of track candidates to be fit (time/fit $= \sim$300 ns); the number of required fits (combinations) is the product of the numbers of hits in each silicon layer. As the hit density in the silicon detectors increases, the number of fits become quite large.

The AM upgrade aims to reduce the number of fits. To achieve it, the width of roads is reduced by increasing the total number of roads per azimuthal wedge from 32×10^3 (32 k) to 512×10^3 (512 k).

The increased bank size can be used not only to reduce the road width, but also for other purposes.

- To increase the track reconstruction efficiency, in particular for tracks with large impact parameter (>1.2 mm, the limit used before the upgrade) or lower the transverse momentum threshold (2 GeV before the upgrade);
- To improve the purity of some triggers, we can increase the SVT use as tracker. We plan to use silicon-only tracks provided by SVT to provide a muon trigger in the forward/backward region, where the XFT coverage is poor.

The SVT upgrade was built in a very short time, since it is designed to require a minimum of new hardware. The new AM++ boards themselves have been under design for a while based on R&D work, the Fast Track processor (FTK) [6], that was done for the Large Hadron Collider (LHC) at CERN. The AM++ board has additional features to interface and control the Amchip03 and for diagnostics. However the AM++ has a smaller input bandwidth compared to the FTK. In FTK, in fact, six input buses are provided for hits coming from the detector. For a six-layer pattern recognition, each layer has a private bus. In the SVT structure, instead, hits from all the layers are multiplexed on a single bus and the CDF AM++ board complies with this standard.

II. Hardware Description

The AM++ is a 9U VME board working at a clock frequency of 40 MHz. It has a modular structure, consisting of four smaller boards, the Local Associative Memory Banks (LAMBs). Each LAMB contains up to 32 AMchips, 16 on each side of the board. The SVT upgrade uses only the 16 AMchips on the top face

Manuscript received November 10, 2005; revised March 13, 2006. This work was supported by INFN.

A. Annovi and S. Torre are with INFN Laboratori Nazionali di Frascati, I-00044 Frascati (Rome), Italy.

A. Bardi, R. Carosi, P. Giannetti, M. Piendibene and F. Spinella are with INFN Sezione di Pisa, 56127 Pisa, Italy (e-mail: paola.giannetti@pi.infn.it).

M. Bitossi, M. Dell'Orso are with INFN Sezione di Pisa, 56127 Pisa, Italy. They are also with Dipartimento di Fisica, Università di Pisa, 56127 Pisa, Italy.

P. Giovacchini, L. Sartori, and B. Simoni are with STMicroelectronics Srl, 20041 Agrate Milano, Italy.

I. Pedron is with Dipartimento di Fisica, Universita' di Ferrara, 44100 Ferrara, Italy. He is also with Atmel Roma s.r.l., 00155 Roma, Italy.

F. Schifano and R. Tripiccione are with INFN Sezione di Ferrara, 44100 Ferrara, Italy. They are also with Dipartimento di Fisica, Universita' di Ferrara, 44100 Ferrara, Italy.

Digital Object Identifier 10.1109/TNS.2006.875355

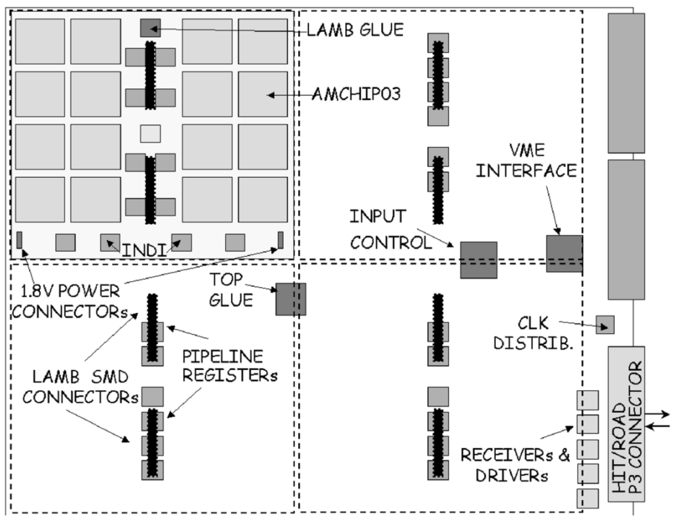

Fig. 1. AM++ board.

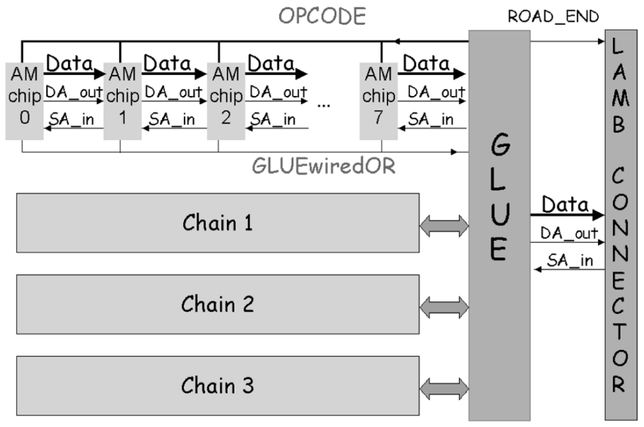

Fig. 2. Device organization on the LAMB.The GLUE chip collects roads from four AMchip pipelines.Each chain can allocate a maximum of eight chips.Roads from the four chains are merged in a single output sent to the TOP GLUE on the AM++.

of the LAMB, which is enough to reach the required pattern density (512×10^3 patterns per wedge). The AM++ board is sketched in Fig. 1. The structure of the LAMB is also shown. The AMchips come in PQ208 packages and contain the stored patterns (memory), pattern matching and read-out logic [4].

When an AM++ board starts to process an event, the hits are received by the Input Control chip. Hits of different layers, arriving on the P3 connector (see Fig. 1), are serialized on a single bus, as required by the SVT architecture. However, inside the Input Control chip, they are demultiplexed again into six buses, one per layer, and simultaneously sent to the four LAMBs. The LAMBs maintain the larger input bandwidth realized in the FTK project, so they may be used also in more complex future applications.

As soon as hits are downloaded into the LAMBs, the locally found roads trigger a request to be read-out (Data Available). Once the event hits are completely read-out, the LAMBs make the last matched roads available within a few clock cycles.

The outputs of the four LAMBs are multiplexed into a single bus by a purposely-designed TOP GLUE chip sending roads through the P3 connector to the Associative Memory Sequencer (AMS), that is the interface between the AM++ and the rest of the SVT processor.

The TOP GLUE, strobed by the AMS, distributes to the LAMB GLUEs a set of Operation Codes (OPCODEs) that define the road read-out order. The majority logic available in the AM chip is driven so that found roads are ordered giving priority to the most constrained matches.

A. LAMB

Six hit buses, one for each detector layer, are fed in parallel to the four LAMBs and distributed to the 32 AM chips on the LAMB through 12 fan-out chips called Input Distributors (INDI). For the output road address bus, the AM chips on each LAMB are divided into four pipelines, each corresponding to a row in Fig. 2, consisting of eight chips, four on the front and four on the back. Each AM chip receives an input bus where road addresses found in previous chips are propagated, multiplexes them with the road addresses internally found in that chip, and sends the output to the next chip. Signals propagate from one chip to another on the top and bottom PCB planes through very short pin-to-pin lines, without any vias.

The outputs of the four pipelines are then multiplexed into a single bus by a specially designed GLUE chip on each LAMB.

The LAMB board has been designed, optimized, simulated, placed, and routed with Cadence software [7]. It represents a significant technological challenge [6] due to the high density of chips allocated on both sides of the board and the use of the advanced chip-scale packages for the 12 INDI chips and the GLUE chip.

Two power voltages are distributed to the LAMBs for the AMchips that require 1.8 V for the core and 3.3 V for the I/O. The 3.3 V is provided, toghether with signals, through the surface mounting connectors placed in the LAMB center, while the 1.8 V is provided through dedicated power connectors placed at the LAMB bottom angles (see Fig. 1).

The LAMB was ready well before the AMchip prototype.It was assembled and tested with FPGAs having a pinout compatible with the AMchip, in order to have in advance a reliable board for the AMchip prototype test. The used FPGA chip [6] (Xilinx Spartan 0.35 μm process) is small and contains an array of 28×28 Configurable Logic Blocks (CLBs).The FPGAs were downloaded with a VHDL code logically equivalent to the AMchip code, reduced to handle only two patterns per FPGA (as opposed to $\sim$5000 patterns for the AMchip). Successful operation has been tested at a clock frequency of 40 MHz.

III. AM++ CONTROL

The internal AMchip [4] configuration specifies a default matching criterion that is activated at the beginning of each event. The matching criterion is controlled through two main parameters, constraining the pattern matching algorithm: *THR* and *required_layers* [4]. THR is the number of layers that are required to be matched in a road in order to get a *road match*. The required_layers option is a 2-bit word. Each bit activate the option for one out of two predefined layers. The road match can be generated only if all active required_layers have a match.

We use the required_layer flag to force the matching of the XFT layer.

The matching criterion can be changed during the event processing through the OPCODEs. The following is the list of operations the GLUE executes for each event, in response to the OPCODEs received from the controller.

1) *Init*: The AM++ and all LAMBs are reset just after the end of previous event (the AM++ controller strobes the Init function). The default matching criterion is loaded (THR and required_layers default values).
2) *Input* (event hits loading): Hits are received from the P3 connector and fed to INDI chips to be propagated to the AMchips.
3) *Decrease Threshold.*: it is executed when the TOP GLUE receives the End-of-Hit from the AM++ controller. When the roads corresponding to the current matching criteria are all read-out, the Decrease Threshold OPCODE is distributed to the AMchips in order to change the value of the THR parameter.
4) *Output* (road read-out): roads can be read-out during or just after the *Input* phase. The matching criterion must not be changed (i.e., OPCODEs must not be sent to AMchips), until the last incoming hit has arrived.

The SVT AM board before the upgrade used the same bus for *Input* and *Output* operations, while the AM++ board provides two independent buses allowing the *Input* and *Output* functions to overlap. With this option the last road leaves the LAMB few clock cycles after the last hit gets in. In fact roads fired in all layers (5/5: 5 matches for five silicon layers) can be collected from the AMchip pipelines as soon as they are available. They do not need to wait until the *input* phase is finished. Roads are thus partially processed even before the event read-out is complete. Roads characterized by a small inefficiency (4/5: 4 matches for five silicon layers), instead, are collected outside the system only when the *input* phase is finished. Roads can be ordered (5/5 first, 4/5 second) or can be read-out without order, at the end of the *input* phase. The ordered read-out is obtained setting the default THR parameter to 6 (five silicon layer matches + XFT match) and decreasing the THR parameter to 5 when the 5/5 roads have been exhausted and the *input* phase is finished. If an unordered read-out has to be reproduced, the order is destroyed setting the default THR to 7 out of six layers, and decreasing it directly to five when the End-of-Hit is received from the controller.

IV. LAMB GLUE Logic

Fig. 2 shows how the different devices are organized on the LAMB. The GLUE chip collects roads from four AMchip chains and controls the matching criterion (all the control signals shown in the figure are active low).

The GLUE chip logic is organized to move roads along the pipelines as soon as possible to reduce latency. The AMchip pipelines can be so long (up to eight chips) that a road matched in a chip far from the GLUE would take many clock cycles to be collected outside the LAMB. For this reason, the GLUE has four independent engines to control the four pipelines in parallel. If a pipeline is ready to change matching criterion, it can

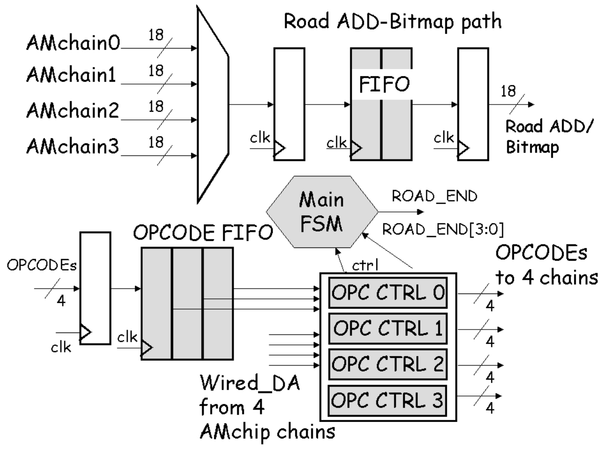

Fig. 3. GLUE logic.

be instructed to do so, even if other pipelines are still working on the default criterion. This makes the GLUE chip more complex (roads belonging to different criteria cannot be mixed in the GLUE output), but helps to build a continuous stream of roads up to the P3 connector.

On each data bus, data are organized into packets of two words: the road address (Road-ADD), and the bitmap, the list of the matched superstrips inside the road. This is a six bits word, one per layer, that contains information about the fired layers: a bit equal 1(0)means that the corresponding layer has (has not) fired.

Packets are pushed ahead in the AMchip chains (see Fig. 2), down to the GLUE. The transfer is controlled by two handshake signals between two adjacent chips: the *Data Available* (DA_out) from the upstream chip to the receiving one and the *Space Available* signal (SA_in) sent back by the receiving chip to the upstream chip.

The protocol is the following: a two word packet transfer is started when both the DA_out and the SA_in signals are asserted. The DA_out signals the availability of a couple of words. The SA_in signals the availability of enough space to receive the two words. The first word (Road-ADD) is latched in the input register on the clock rising edge on which both DA_out and SA_in are asserted. The second word (bitmap) is latched on the next clock cycle regardless of the state of control signals.

A simplified scheme of the GLUE chip is shown in Fig. 3. The first purpose of the GLUE is to multiplex into a single stream the four AMchip-chain buses. The upper part of Fig. 3 shows that after the multiplexer there is a register, then a two word FIFO and finally the output register. This means that each word needs at least three cycles to go from input to output. The controlling Finite State Machine (FSM) makes sure that packets composed of two words (Road-ADD, bitmap) are not broken.

The second purpose of the GLUE is to send OPCODE lines to all the AMchips on the LAMB (see Fig. 2). Four buses are provided by the GLUE, one for each AMchip chain. All the AMchips in a chain receive the OPCODEs in parallel. OPCODE and wired_DA (wired Data Available) lines allows the GLUE to control the AMchip matching criteria. Each AMchip has an output called wired_DA. It is active if the AM chip has at least one road available for readout. The eight wired_DA signals in a chain are wired together into the GLUEwired_DA signal, providing the logic OR of all wired_DA pins in a chain.

The GLUEwired_DA provides fast information to the GLUE on the availability of roads inside each of the four chains (eight AMchips per chain). The GLUE logic must take care of the fact that the wired_DA signal becomes active with some clock cycles of delay with respect to the incoming hits or OPCODEs that actually generates the road match. The GLUE guarantees that roads extracted in correspondence with different OPCODEs are not mixed.

The GLUE logic informs the TOP GLUE using the Road_end signal as soon as all found roads for a given criterion have been read-out.

The lower part of Fig. 3 shows the OPCODE logic. The OPCODEs coming from the TOP GLUE go through a register and then reach the Opcode FIFO. This FIFO stores all OPCODE words (up to three) that have not been sent to all chains yet. There are four "OPC CTRL" blocks working in parallel, one for each AM chain. All three registers of the Opcode FIFO are accessed in parallel by the "OPC CTRL" blocks. Each block sends OPCODEs to one chain of AMchip. The Main FSM coordinates the four blocks.

If an OPCODE is made of two words (OPCODE word + DATA word), the "OPC CTRL" block identifies those packets and avoids the separation of words.

Every OPCODE is sent (written into the output register) to the chain only when the previous OPCODE has been completely processed. An OPCODE is considered processed when all roads generated by that OPCODE have been extracted from the AMchip chain. This condition is verified checking the AND of these two conditions: 1) at least N clock cycles elapsed since the OPCODE has been sent, where N is the number of cycles necessary for the OPCODE to activate the GLUEwired_DA signal (in case of found roads); 2) no more roads are left in the chain. This is verified checking that the GLUEwired_DA signal is inactive.

When the OPCODE is completely processed, a Road_end signal is sent from the "OPC CTRL" to the "Main FSM" and the next OPCODE, if any, is sent to the chain. The purpose of the Road_end signal is to tell the "Main FSM" that roads for the current OPCODE are finished.

With regards to OPCODEs the "Main FSM" has three tasks to perform: 1) make sure that roads extracted with different OPCODEs are not mixed together; 2) propagate a Road_end signal, when all the four chains have exhausted their roads, to the TOP GLUE on the AM board; 3) remove from the FIFO the OPCODEs already processed by all four chains.

In order to accomplish the first task the "Main FSM" holds (it does not produce the Space Available signal) all data streams but those coming from chains that are processing the earlier OPCODE. For the second task it will send the Road_end signal only after the last road corresponding to an OPCODE has left the GLUE.

V. Board Configuration and Diagnostics

All the logic functions of the AM++ board are implemented on two kind of chips, Xilinx complex programmable logic devices (CPLDs) [8] and standard cell AMchips. For the programmable logic we selected mainly 3.3 V devices with 5 V I/O compatibility since the AM++ board logic is connected to the VME 5 V transceivers. These devices implement part of the VME slave functions and provide I/O voltage translation for the other logic on the board. The low-cost XC9500XL chips were used for the PIPELINE REGISTERS and the INDI chips, placed just before and after the LAMB connectors (see Fig. 1), while for the control logic placed in the Input Control, TOP GLUE and VME chips, we have chosen the 3.3 V most powerful device of the CoolRunner XPLA3 family (XCR3512XL). Since the AMchips on the LAMB require also the 1.8 V power supply for the core (in addition to the 3.3. V for I/O), we have chosen for the LAMB GLUE the most powerful device (XC2C512) of the 1.8 V CoolRunner II family (the LAMB GLUE has no direct contact with the VME buses).

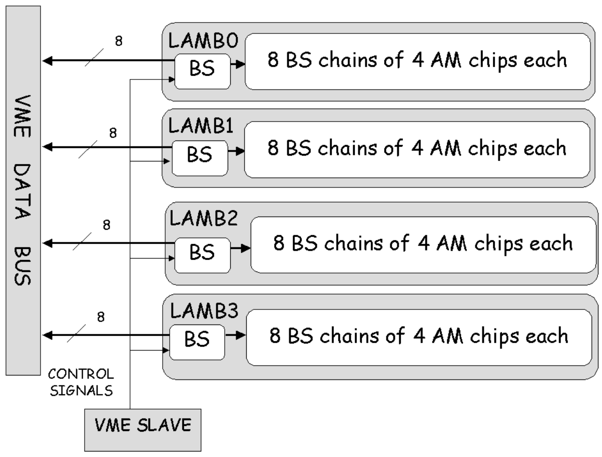

Fig. 4. Daisy chains of AMchips on the LAMB. The 32-bit wide VME data bus gives access to the 32 chains in parallel. The custom VME Slave generates all the control signals necessary to correctly operate on the BS chip interface (located on each LAMB) controlling all the AMchip Jtag signals.

All chips on the AM++ and the LAMB boards are fully configurable and testable using the Jtag interface implemented in the devices, following IEEE Std. 1149.1-1990 [9].

Jtag circuitry includes a standard interface through which instructions and test data are communicated between chips and a host, making it possible to download a bitstream to the chips to configure their internal logic, as well as to setup its I/O pins to specific logic levels through Boundary Scan Register. Conversely it is possible to upload a bitstream from the chip to gather information about logic level sampled at its I/O pins.

CPLDs used in the AM++ board are commercially available chips whose internal architecture is fully configurable accessing the built-in Jtag Interface. The configuration is retained in a flash memory inside the chip, which preserves them from loosing configuration upon power down. Due to the serial nature of Jtag data flow, several chips can be connected in daisy chain, and accessed for individual configuration and test. With a download cable connected to a PC and programming software the serial bitstream can be downloaded to the chips accessing dedicated connectors on the AM++.

The Amchip03 devices implement a Jtag interface as well [4]. This interface gives access to internal configuration of the AMchip which includes pattern memory and internal registers Writing/Reading as well as Boundary Scan logic.

AMchips are interconnected on each LAMB in eight separate daisy-chains of four devices each, to build a total of 32 individual daisy-chains throughout the whole AM++ (see Fig. 4) The 32 separate chains are accessed in parallel by a VME slave which gives access to all Jtag control signals of the chains, in parallel. The VME 32-bit wide data transfer allows us to program the 32 chains in parallel, so that time needed to store Patterns in the AMchips is a few seconds. Patterns are also checked using the boundary scan.

Very useful Boundary Scan diagnostics have been developed. These diagnostic tools are based on the use of two Boundary Scan instructions, SAMPLE/PRELOAD and EXTEST.

The SAMPLE/PRELOAD instruction allows us to take a sample of the functional data entering and leaving the device while the device remains in its functional mode. This is useful to sample logic state of I/O pins. This instruction is also used to preload test data into a dedicated internal boundary-scan register, prior to loading an EXTEST instruction. During the EXTEST instruction, the boundary scan cells associated with outputs are preloaded with test patterns to test downstream devices. The input boundary cells are set up to capture the input data for later analysis.

We developed an important diagnostic tool taking advantage of the boundary scan instructions specified above and a special "Test Mode" implemented in the AM++. In this operating mode it is possible to write/read hit and road data from the VME slave to the Input Control and TOP GLUE chips respectively, substituting the P3 connector and LAMB normal sources of data. The Input Control chip sends the "test hits" to the four LAMBs. Hit data flow to the INDI chips and are then sent to all the AMchips, in parallel. All AMchip pads are read-out to check if the downloaded values have been correctly transmitted. In this way, we could find interconnection problems in the hit flow. A similar procedure is available to check the road flow from the TOP GLUE to the P3 connector. These tests have been performed on all the produced AM++ and LAMB boards so assembly problems could be detected during the first step of the validation procedure (see below). These tests allowed to quickly find and solve most of the assembly problems.

VI. Board Validation

The commissioning of the AM++ board took place in the summer of 2005 while the experiment was taking data. A very careful test procedure was devised to reduce to a negligible level the impact of the commissioning on detector operation and functionality. The test steps are described below.

1) The first validation was performed on a stand-alone test stand, where random events were sent through the new boards and the output was compared to board level simulation.
2) A second level of validation was performed using real data spied from the experiment. We used a test crate, placed near the real SVT, to reproduce the configuration of an SVT piece based on a single AM board. We split the inputs to one SVT old AM board and sent a copy to the new AM++ in the test crate. The output of the crate could be compared directly with that of the SVT AM receiving identical input data. We proceeded in the installation only after all discrepancies between the two systems were completely understood. The same inputs were also fed to a board-level simulator. Comparison between the hardware and the simulation was used to validate both the board and the upgraded simulation itself.
3) After successfully passing step 2, one new AM++ board replaced an old one for a short, low-luminosity test, and, after being successful, for 100 hours of data taking at any luminosity. This final test was important because it provided higher statistics than the previous tests, allowing for detection and debugging of lower rate errors. Furthermore, it was an extra check that the control signals used by the DAQ system [10] do not interfere with the board functionality. During these tests the standard SVT monitoring, which runs on crate processors, was used for validation. We monitored the impact parameter, the angle in the transverse plane and the track transverse momentum distributions.
4) At this stage, SVT tracks were compared with simulation. The efficiency and failure rates were monitored. The efficiency is defined as the fraction of SVT tracks matching more precise and complete track reconstruction and is measured to be $\sim 80\%$. The failure rate is defined as the fraction of reconstructed tracks that do not match to simulated tracks. The failure rate must be as low as possible for two reasons: 1) failures can be a symptom of hardware problems, and 2) we need the simulation to reproduce the hardware in a very detailed way for purposes of data analysis, where the simulation is used to understand various efficiencies. We allow for a failure rate of the order of 10^{-3} in the whole SVT.

The AM++ had to work correctly in data taking at both low and high instantaneous luminosities before it was considered ready to be installed into all 12 wedges.

VII. Conclusion

A new associative memory board has been produced for the SVT upgrade in the CDF experiment. This board provides an improvement of a factor ~ 40 in the size of the trajectory bank, which for SVT application translates to a road size three times smaller. The AM++ has been succesfully tested, installed and used in CDF, running at 40 MHz, a 30% clock rate improvement over the SVT system before the upgrade.

The AM++ helps to improve the data acquisition capabilities within CDF. The SVT processing time has been significantly reduced for complex events. As a consequence, the trigger bandwidth has been increased by$\sim$5 kHz [11].

The speed and flexibility of the new SVT allows for the implementation of better algorithms to further speed up the system and improve tracking performance.

References

[1] M. Dell'Orso and L. Ristori, "VLSI structures for track finding," *Nucl. Instrum. Meth.*, vol. A278, pp. 436–440, 1989.

[2] R. Amendolia, "The AMchip: A full-custom CMOS VLSI associative memory for pattern recognition," *IEEE Trans. Nucl. Sci.*, vol. 39, no. 4, pp. 795–797, Aug. (1992).

[3] A. Bardi, "A programmable associative memory for track finding," *Nucl. Intsrum. Meth.*, vol. A413/2-3, pp. 367–373, 1998.

[4] A. Annovi, “A VLSI processor for fast track finding based on content addressable memories,” in *IEEE Nucl. Sci. Symp. Conf. Rec.*, Oct. 23–29, 2005, vol. 1, pp. 259–263.

[5] W. Ashmansk, “The CDF silicon vertex trigger,” *Nucl. Intsrum. Meth.*, vol. A518, pp. 532–536, 2004.

[6] A. Annovi, “A pipeline of associative memory boards for track finding,” *IEEE Trans. Nucl. Sci.*, vol. 48, no. 3, pp. 595–600, 2001.

[7] *Cadence software*, [Online]. Available: http://www.cadence.com/

[8] *The Programmable Logic Company 2001 Feb*, [Online]. Available: http://www.xilinx.com/products/silicon_solutions/index.htm, Xilinx Xilinx Data Book 2000. Xilinx. San Jose, CA.

[9] *IEEE Std. 1149.1-1990 IEEE Standard Test Access Port and Boundary-Scan Architecture Description*, [Online]. Available: http://standards.ieee.org/reading/ieee/std_public/description/testtech/1149.1-1990_desc.html

[10] A. Belloni, “Event builder and level 3 at the CDF experiment,” *Nucl. Instrum. Meth.*, vol. A518, pp. 522–524, 2004.

[11] J. Adelman, “First steps in the silicon vertex trigger upgrade at CDF,” in *IEEE Nucl. Sci. Symp. Conf. Rec.*, Oct. 23–29, 2005, vol. 1, pp. 603–607.