A VLSI Processor for Fast Track Finding Based on Content Addressable Memories

A. Annovi, A. Bardi, M. Bitossi, S. Chiozzi, C. Damiani, M. Dell'Orso, P. Giannetti, P. Giovacchini, G. Marchiori, I. Pedron, M. Piendibene, L. Sartori, F. Schifano, F. Spinella, R. Tripiccione

***Abstract*– We describe a VLSI processor for pattern recognition based on Content Addressable Memory (CAM) architecture, optimized for on-line track finding in high-energy physics experiments. A large CAM bank stores all trajectories of interest and extracts the ones compatible with a given event. This task is naturally parallelized by a CAM architecture able to output identified trajectories, recognized among 2^{96} possible combinations, in just a few 40 MHz clock cycles. We have developed this device (called the AMchip03 processor) for the Silicon Vertex Trigger (SVT) upgrade at CDF using a standard-cell VLSI design methodology. This approach provides excellent pattern density, while sparing many of the complexities and risks associated to a full-custom design. On the other hand, the cost performance ratio is well more than one order of magnitude better than an FPGA-based design. This processor has a flexible and easily configurable structure that makes it suitable for applications in other experimental environments. We look forward to share this technology.**

I. Introduction

TRACK reconstruction in high energy physics experiments requires large online computing power.

A conceptually very simple and effective approach to the problem, proposed several years ago [1], can be applied to a large class of detectors, e. g. typical vertex detectors, that can be conceived as segmented in layers, each layer being in turn divided into a number of bins. For each event a number of particle tracks traverse the detector. Each track crosses one bin per layer, generating hits. Tracks are therefore associated to specific strings of hits and misses (that we code as 1's and 0's respectively). All tracks of physical interest correspond to bit patterns that are explicitly enumerated and stored in an appropriate data bank. Track reconstruction in this approach amounts to retrieve from the data bank the string (or strings) that match the current event, as seen by the detector. This task can be performed with negligible time delay by a Content Addressable Memory (CAM), i.e. a device that compares in parallel an input pattern with all stored patterns and return the address(es) of the matching location(s).

A full-custom VLSI technology was used in this context several years ago for the CDF experiment, where tracks inside very high multiplicity events must be found in a time span of a few microseconds. A fast online tracker for the CDF silicon vertex detector [2] was built around the purposely developed Associative Memory (AM) chip [3].

A critical figure of merit for a CAM-based track reconstruction system is the number of patterns that can be stored in the data bank. In the past, the request to maximize available patterns forced a full-custom VLSI approach, which implied a big development effort and a difficult upgrade path to more recent and denser micro-electronic technologies, as they eventually become available. In recent years, very high-density silicon technologies make it possible to build a very large number of transistors inside a reasonably large silicon area (say, 1 cm^2). It is therefore appropriate to reconsider the best trade-off between pattern density and ease of design (and eventually re-design). While the full-custom approach obviously maximizes pattern density, an FPGA-based design gives the fastest development time at the cost of a drastically reduced pattern density. This option has been considered in [4]. Midway between the two approaches, a standard-cell based design brings substantial advantages, as discussed in details later on.

In this paper, we describe the design and test of a new much more powerful version of the AM chip, that we call AMchip03 processor, now used for the upgrade of the Silicon Vertex Trigger (SVT) at CDF. The AMchip03 uses a 0.18 micron CMOS technology and a strictly standard-cell based VLSI design approach.

This paper is structured as follows: in the next section we describe the architectural features of the new processor. We then cover implementation and test issues, including a discussion of the relative merits of the various VLSI design approaches. Finally, we present our conclusions, including long term perspectives for this CAM-based approach to track identification and reconstruction

Manuscript received November 10, 2005. (This work was supported by INFN).

A. Annovi, A. Bardi, M. Bitossi, M. Dell'Orso, P. Giennetti, P. Giovacchini, M. Piendibene and F. Spinella are with Istituto Nazionale di Fisica Nucleare, Largo Pontecorvo 3, 56100 Pisa, Italy (telephone: +39-050-2214000, e-mail: firstname.lastname@pi.infn.it).

A. Annovi, M. Bitossi, M. Dell'Orso, P. Giovacchini are with Dipartimento di Fisica, Università di Pisa, 56100 Pisa, Italy (telephone: +39-050-2214000, e-mail: firstname.lastname@pi.infn.it).

S. Chiozzi, C. Damiani, I. Pedron, L. Sartori, F. Schifano and R. Tripiccione is with Dipartimento di Fisica, Universita' di Ferrara, 44100 Ferrara, Italy.

S. Chiozzi, C. Damiani, L. Sartori, F. Schifano and R. Tripiccione are with INFN Sezione di Ferrara, Ferrara, Italy.

G. Marchiori and I. Pedron are with Atmel Roma, Roma, Italy.

II. AMCHIP03 ARCHITECTURE

The AMchip03 is a single chip digital VLSI processor performing all functions needed for CAM-based fast track identification and reconstruction, to format and compress the results and to pass them to a daisy-chain of further AMchip03 components and eventually to the downstream stages of the data readout system.

As already briefly sketched in the previous section, a large bank of precalculated patterns is used to find tracks in a detector consisting of a number of layers, each layer being segmented into a number of bins. For each physical event a number of tracks traverse the detector. Each track crosses one bin per layer, generating hits. The collection of tracks creates a particular configuration of hits: we use the word event when referring to the full set of hit bits. We know which bins have been hit and from this information we want to reconstruct the trajectories of all particles.

To this purpose we list offline, once and forever, all the relevant tracks that can go through the detector. A hit pattern, consisting of one hit bin per layer, corresponds to each possible track. All the different hit patterns are stored in a sufficiently large memory with CAM capabilities, that we call the pattern bank. For each event one has to scan the pattern bank and compare each pattern to the event. A track candidate is found whenever all the hits in the pattern, or a majority of them, are present in the event. The CAM technology, that allows performing this search in just a few clock cycles, is obviously the key component of the AMchip03 (in the following we will also loosely refer to the CAM bank with the term "Associative Memory").

It must be emphasized that with respect to commercially available CAM's the AMchip03 has the unique ability to search for correlations among input words received at different clock cycles.

The Associative Memory allows comparing at the same time all the previously stored patterns with the current event. The AMchip03 performs pattern recognition in a detector of up to 6 layers. Each pattern is therefore segmented into six 16 bit words, one for each detector layer. Fig 1 displays a block diagram of the complete device. The chip receives 6 buses (Bus0-5 on the left of fig. 1) bringing hits from the detector, one bus for each layer. At each clock cycle the content of each pattern word is compared in parallel with the content of the corresponding bus and a flip-flop is set for each occurring layer match. When the number of layer-matches for a certain pattern is above a given threshold, a pattern-match flip-flop is set. Threshold comparison is introduced to account for any inefficiency in the detectors. It is possible to require that one specific layer has fired, in order to set the match flip-flop. The pattern bank uses approximately 80% of the silicon resources in the device and contains 5120 patterns, corresponding altogether to approximately 500,000 content-addressable memory bits.

In addition to the default 6 layer mode, the chip can be configured to perform pattern recognition for a detector of up to 12 layers, combining each pair of 6 word patterns into a 12 word pattern.

The processor contains the logic functionalities needed to interface with other sections of the readout system. The addresses identifying all matched pattern are read-out from the chip through the Patt_add_out bus. Patterns are read-out one after the other, giving priority to the highest address. When a pattern is read-out, "kill" logic resets the "pattern-match" flip flop until the end of the event, so each matched pattern is read-out once.

Several AMchip03 processors can be cascaded in daisy chain to build a larger pattern bank. Multiple daisy chains can in turn be multiplexed for even larger banks. Each AMchip03 outputs its matched addresses on the Patt_add_out bus. Pattern addresses can be received on the Patt_add_in bus by the following AMchip03 in the pipeline. These addresses are stored in a FIFO and multiplexed with local addresses on the only output, Patt_add_out. The flux of addresses is controlled by a finite-state-machine (Data Flux Control in the figure) that uses a simple protocol based on 4 control signals, labeled as DA_out, DA_in SA_out and SA_in in the figure.

The AMchip03 is initialized by writing all locations on the pattern bank and a number of configuration registers. All initialization and book-keeping operations are performed via JTAG interface.

The core of our architecture is similar to the old AM device, previously used in CDF [3], but it has a much larger array size. On the other hand, this design substantially improves input-output capabilities, critical to reduce processing time. A short processing time is an important asset, since it helps reduce the trigger dead-time. The time associated to actual pattern matching is negligible once data is available inside the chip, since the CAM array identifies a matching pattern in just one clock cycle. Hence, total processing time is determined essentially by input bandwidth. In a typical CDF situation, ~300 x 12 hits are recorded by 12 AMchip03 based systems working in parallel. Due to system constraints, all hits are loaded into the chip over just one bus (40 MHz rate) at a time, so input time is ~ 7.5 μsec. Processing is already active as data is input to the processor, and matched patterns are immediately offloaded, with the last pattern coming out approximately 6 clock cycles (150 ns) after the last input data is received. In a more demanding environment (we consider a typical LHC situation, with ~30000 hits in the whole detector), we might expect ~2000 hits going to one AMchip03 array handling for instance $1/16^{th}$ of the whole detector. Exploiting the full input bandwidth (6 busses), input time in this case is of the order of 8 μsec.

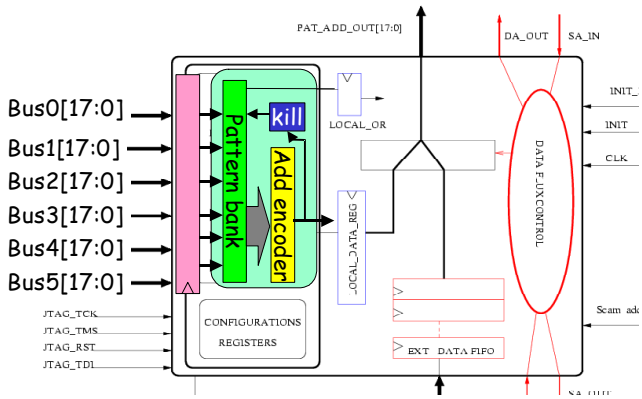

Fig. 1. Block diagram of AMchip03. The readout and control logic is emphasized (see text for description). While the pattern bank, which occupies 80% of the device area, is represented by the left most rectangle.

III. Implementation and Tests

The AMchip03 has been logically designed using the VHDL hardware description language, and has been mapped on a standard-cell library with Synopsys EDA tools, using the UMC 0.18 micron CMOS process (with 1 poly and 6 metal (Al) layers). The automatically synthesized pattern array has been then further optimized manually trying to reduce logic complexity and with an eye to regular placement & routing. The CAM bank is the part of the project that occupies most of the area so it is important to exploit as much as possible the intrinsic regularity of the array. The main goal at this stage is gate density, since our speed goals (40MHz – 50MHz clock frequency) are easily obtained with the available silicon technology. Only a manual study of the structure can guarantee this result. Once an array structure reasonably close to optimal is obtained with a few trial and error steps, the associated cell placement is coded into a relatively simple Perl script. The remaining logic has been handled by standard EDA tools: by Synopsys Physical-Compiler for placement of all other modules of the processor and Cadence First-Encounter for routing. The die size is approximately 9.8 x 9.8 mm^2. The chip core uses approximately 80% of the available area. The chip has a total of 164 logical input and output pads and is packaged in a standard 208-pin plastic "flat-pack". Fig. 2 shows the micrograph layout of the AMchip03.

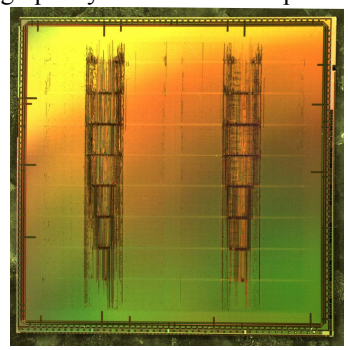

Fig. 2. Micrograph of the AMchip03 device. Four manually optimized columns of 1280 patterns each are visible. One on the left, one on the right and two in the middle. The two columns of lower density logic correspond to the interconnection and readout logic that was automatically placed.

We came to the common problem of chip design functionality verification against specification. This problem was separated into two logical steps. First take advantage of the fact that standard-cell designs are described by detailed and reliable Verilog models to be used for simulation. Second compare device simulation with appropriate reference test vectors that comply with, and verify, functional specifications. In order to prepare the reference test vectors, we independently developed a C-language model of the AMchip03. We wrote a total of 200Kbytes of code. A set of input test vectors was randomly generated, in order to load random patterns and random input data. Another set is made of manually defined test vectors used to test all specific configurations.

Both the C and Verilog model have been extensively simulated prior to fabrication with those inputs. Finally the output of the two models have been compared and debugged in order to understand and remove all discrepancies. As result we had a fully debugged design, a clarification of all misinterpretations of the specifications as well as a working device with the first prototype.

Post-routing simulations and static timing analysis shows a maximum operating frequency marginally higher than 50MHz in worst case conditions.

Prototypes of the device were fabricated in a Multi Project Chip (MPC). A pilot production run of 12 wafers (~3000 chips) with dedicated masks was performed immediately after. While the measured yield on the MPC has been disappointingly low (~37%), the dedicated fabrication run had a yield (~70%) marginally better than expected for our die size. The first pilot run was sufficient to produce the number of chips necessary for the CDF upgrade (~2000 chips).

We have developed a test board with a Zero Insertion Force (ZIF) socket to test the processor before soldering them on the application boards. The test board has connectors for Pattern Generator and Digital Analyzer. FIFO's are used to handle the large amount of input data (6 hit buses, 18 bits for each bus). The prototype standard cell chips have been extensively tested with the same test vectors used to validate the design trough simulation. Software to convert automatically the test vectors from the simulation to the test stand environment has been developed. Chips selected by the test stand have been soldered on boards and tested on the experiment, showing that the selection performed in the single chip test stand is 100% efficient. Chips have been tested up to a frequency of 40 MHz, the maximum working frequency in CDF.

The AMchip03 pinout is compatible with an FPGA chip, so that extensive board tests have been prepared before receiving our prototype. We have used a commercial low cost FPGA family (Xilinx Spartan 0.35 micron process) [5]. The FPGA-based AMchip03 mock-up has been logically designed with the same VHDL code that defines the real device. As expected from the discussion above, the pattern count in the FPGA is drastically reduced with respect to the full AMchip03 implementation (only 2 patterns, instead of 5120).

For SVT, 9U VME boards housing 4 mezzanines are used. Each board uses up to 128 chips, corresponding to 640K patterns [8], although only 64 chips are used in the currently available system. The complete system has 24 boards (that is, 1536 AMchip03 parts), handling the 12 wedges of the detector.

The measured currents, I_{core} and I_{IO}, drawn by the AMchip03 from respectively 1.8 V and 3.3V power supplies are shown in Table I. Measurements have been done in worst-case conditions when all input buses switch simultaneously.

TABLE I
Power consumption of AMchip03

Frequency (MHz)	I_{core} (mA)	I_{IO} (mA)
10	200	40
20	400	80
40	700	160

It is interesting to compare the performances of our processor with other possible approaches. Let us start with a comparison with an FPGA-based implementation. The most critical resources for a CAM structure are storage elements that can be accessed all in parallel. In a typical FPGA, these storage elements can only be implemented with random flip-flops. A typical advanced FPGA family (for instance, Altera Stratix 2 [6]) has between 12000 and 140000 flip-flops. Each patterns requires of the order of 100 flip-flops, so between ~ 100 and ~1500 patterns could be squeezed inside one FPGA, under the very optimistic assumption that 100% usage as well as satisfactory speed can be obtained at the same time. Taking the largest FPGA option as an example the overall chip inventory (and the corresponding engineering problems) would increase by a factor 3 to 4, and the overall system costs by an unbearable factor of 30 to 40. The design effort would be probably reduced by a small factor from the 8 months needed for the development of the AMchip03.

At the other technological extreme, a full custom VLSI approach (at least for the regular CAM array) might increase density by a factor of approximately 4, while fabrication costs would stay almost constant. On the other hand the design process would be much longer in time, and re-targeting to a different VLSI technology would be almost equivalent to a full re-design.

Finally, it is interesting to estimate the hardware complexity of a farm of PC's trying to perform our pattern matching task in software with comparable performance. Direct implementation in software of the actual algorithm performed by the Amchip3 is obviously ruled out. Assume that n ~ 200 hits are recorded and that our upper limit on the processing time is 10 μsec. For each event a comparison has to be made with all stored patterns. If each pattern is stored as 4 words in cache (optimistically accessed at a rate of 1 word every 2 nsec), one PC is not able to inspect more than ~1000 patterns in the allotted time frame, so the performance of just one AMchip03 can be equalled by 200 – 400 PC's. N ~ 10^6 patterns could be handled by ~ 40000 PC's, an inordinately large number.

More efficient algorithms are available however. Let us focus first on just one layer of w bits. For each hit on that layer (i.e., one binary word of w bits), we may expect that approximately $N/2^w$ candidate patterns are found. With typical CDF values: n ~ 50 hits/layer, and w = 12 bits and N = 640000, we expect of the order of 160 candidate patterns, that we may pre-compute and insert in a look-up table. If we expect n hits, after looking into the look-up tables, we end up with a list of approximately n $N/2^w$ entries, containing candidate patterns for each layer. L such lists have to be prepared, to cater for all layers. The contents of these L lists must now be inspected, looking for equal content (that is for a pattern identifier that is common to all layers, flagging a good candidate). Efficient algorithms exist to perform the task, with a number of memory accesses of the order of

$$n\frac{N}{2^w}\log_2(n\frac{N}{2^w})$$

This step must be performed L times, implying a number of memory accesses of the order of

$$KL\,n\frac{N}{2^w}\log_2(n\frac{N}{2^w})$$

where K is a small integer K = 2 ... 5.

Using CDF reference values, we end up with approximately 3 10^6accesses. If we optimistically assume that all data is available in a large L2 cache (typical access time optimistically estimated at 2 nsec) and just neglect any processing overhead, we conclude that approximately 300 PC's could perform the task in parallel in order to stay within the allotted time limits. A system able to handle all 12 wedges would therefore include from 3000 to 4000 PC's.

IV. CONCLUSIONS

A track finding processor based on a CAM has been produced for the SVT upgrade of the CDF experiment. This device improves on the size of the pattern bank by a factor ~40 over a previous version of the processor. The device is also faster: it was designed with a target clock frequency of 50MHz (although it is currently operated in a 40 MHz system).

The design uses a standard-cell based technology instrumental to keep the design time to about 8 months (plus 2 months to fabricate the prototypes). Our design methodology makes upgrades to new technologies or to different environments straightforward and quick to implement.

We can extrapolate the future capacity of a new device based on 130-90 nm technologies: we expect an improvement of at least a factor 2-4 in pattern density. A good strategy is to

use FPGAs for system development, and to start migration to a pin compatible standard-cell device only for system production when the experiment is getting close to data taking. Clock frequencies, that in the present application had very relaxed requirements, could also be increased by large factors (~5). This might provide opportunities to use these devices in applications with tighter time constraints, such as Level 1 trigger systems.

We would like to thank C. Das, P. Malisse and D. Lambricht of IMEC (Belgium) and G. D'Amelio of Microtest (Italy) for their precious collaboration in all production and test phases of the AMchip03.

V. REFERENCES

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

[2] S. Belforte et al., "SVT: An Online Silicon Vertex Tracker for the CDF Upgrade", *Nucl. Intsr. and Meth., vol. A409, 1998* pp. 658-661.

[3] R. Amendolia et al., "The AMchip: a Full-custom MOS VLSI Associative memory for Pattern Recognition" *IEEE Trans. on Nucl. Sci., Vol.39, 1992* pp. 795-797.

[4] A. Bardi et al. "A Programmable Associative Memory for Track Finding", *Nucl. Intsr. and Meth., vol. A413/2-3, 1998* pp.367-373.

[5] http://www.xilinx.com.

[6] Altera Corp., Stratix II Device Handbook (2004). See also http://www.altera.com

[7] http://www.synopsys.com.

[8] A. Annovi et al. "The AM++ Board for the Silicon Vertex Tracker upgrade at CDF" N14-116 IEEE 2005 NSS, Puerto Rico, PR, USA, October 22-29, 2005

[9] A. Bardi et al. "A large Associative Memory system for the CDF Level 2 Trigger" N16-2 IEEE 1998 NSS, Toronto, Ontario, CANADA, November 8-14, 1998.