A New Variable-Resolution Associative Memory for High Energy Physics

A. Annovi, S. Amerio, M. Beretta, E. Bossini, F. Crescioli, M. Dell'Orso, P. Giannetti, J. Hoff, T. Liu, D. Magalotti, M. Piendibene, I. Sacco, A. Schoening, H.-K. Soltveit, A. Stabile, R. Tripiccione, V. Liberali, R. Vitillo, G. Volpi

***Abstract*– We describe an important advancement for the Associative Memory device (AM). The AM is a VLSI processor for pattern recognition based on Content Addressable Memory (CAM) architecture. The AM is optimized for on-line track finding in high-energy physics experiments. Pattern matching is carried out by finding track candidates in coarse resolution "roads". A large AM bank stores all trajectories of interest, called "patterns", for a given detector resolution. The AM extracts roads compatible with a given event during detector read-out.**

Two important variables characterize the quality of the AM bank: its "coverage" and the level of fake roads. The coverage, which describes the geometric efficiency of a bank, is defined as the fraction of tracks that match at least one pattern in the bank. Given a certain road size, the coverage of the bank can be increased just adding patterns to the bank, while the number of fakes unfortunately is roughly proportional to the number of patterns in the bank. Moreover, as the luminosity increases, the fake rate increases rapidly because of the increased silicon occupancy. To counter that, we must reduce the width of our roads. If we decrease the road width using the current technology, the system will become very large and extremely expensive.

We propose an elegant solution to this problem: the "variable resolution patterns". Each pattern and each detector layer within a pattern will be able to use the optimal width, but we will use a "don't care" feature (inspired from ternary CAMs) to increase the width when that is more appropriate. In other words we can use patterns of variable shape.

As a result we reduce the number of fake roads, while keeping the efficiency high and avoiding excessive bank size due to the reduced width.

We describe the idea, the implementation in the new AM design and the implementation of the algorithm in the simulation. Finally we show the effectiveness of the "variable resolution patterns" idea using simulated high occupancy events in the ATLAS detector.

ATL-UPGRADE-PROC-2011-004
20 May 2011

Manuscript received May 20, 2011. This work was supported in part by the U.E. IOF project 254410 .

A. Annovi, M. Beretta, and G. Volpi are with INFN Frascati.

S. Amerio is with INFN Padova.

E. Bossini, F. Crescioli, M. Dell'Orso, P. Giannetti, M. Piendibene, I. Sacco, R. Vitillo are with INFN Pisa.

J. Hoff, T. Liu are with Fermilab.

D. Magalotti is with INFN Perugia

A. Schoening, H.-K. Soltveit are with University of Heidelberg

A. Stabile, V. Liberali are with INFN Milano.

R. Tripiccione is with INFN Ferrara.

I. Introduction

Track reconstruction in high-energy physics experiments requires large online computing power. The Fast Tracker (FTK) for the ATLAS trigger [1] is an evolution of the CDF Silicon Vertex Tracker (SVT) [2], [3], the only state-of-the-art online processor that tackles and solves the full track reconstruction problem at a hadron collider.

The FTK track fitting system approaches offline tracking precision with a processing time of the order of tens of microseconds, compatible with 100 kHz input event rates. This task can be performed with negligible time delay by doing pattern recognition with a content addressable memory (Associative Memory, AM), i.e. a device that compares in parallel the event hits with all the stored pre-calculated low resolution track patterns and returns the addresses of the matching patterns. A second processor receives the matching patterns and their related full-resolution hits to perform the final track fitting (Track Fitter, TF).

A critical figure of merit for the AM-based track reconstruction system is the number of patterns that can be stored in the bank. For the SVT upgrade [2], [4] we developed a version of the AM chip (AMchip03 processor) [5] using a 180 nm CMOS technology and a strictly standard-cell VLSI design approach. The AM chip upgrade increased the number of patterns stored in a chip from 128 to 5×10^3 and it could work at a 50 MHz frequency[1].

The FTK processor proposed for the ATLAS experiment is much more ambitious than SVT. In fact a very high efficiency and high quality track reconstruction, already shown possible by SVT, must be achieved in a much more complex detector. Moreover, the higher luminosity ($>10^{34}$ cm^{-2} s^{-1}) will increase the complexity of events. As a consequence a very large bank is necessary: candidate tracks have to be found with more than 95% efficiency over the whole tracking detector ($|\eta|<2.5$), with high efficiency down to transverse momentum of 1 GeV, and the pattern recognition has to be extended to 11 silicon detector layers (3 pixel layers and 8 SCT layers) with a small enough pattern width to reduce drastically the number of fake tracks and the track fitting processing time.

II. The Pattern Bank

As already mentioned, two important variables characterize the goodness of the AM bank: its coverage and the rate of fake roads. The coverage is a purely geometric

[1] Higher frequency could be achieved, but it was not required by the final application.

quantity, defined as the probability that a track (with helix parameters within the desired range) intersects at least (M-1) of the M silicon layers within a pattern in the bank. In short, it is the fraction of reconstructable tracks based only on the detector geometry; it describes the geometric efficiency of the bank only. Track efficiency instead includes the contributions of all the algorithms used in FTK, the clustering of single detector channels before the AM and, after the AM, the track fitting inside roads whose efficiency is determined by a χ^2 cut.

We generate tracks in the whole detector (no detector symmetries are exploited, to prevent alignment problems) and we store new patterns corresponding to the generated tracks, until the bank reaches the desired coverage. Each layer is subdivided into bins (Super Strips, SS) of equal size, each one is a rectangle in φ-z space. A pattern is a combination of M Super Strips, one in each layer. Only patterns that are hit by at least one track are kept. The whole collection of generated tracks is the FTK training sample. The bank is generated using very large training samples produced with the full ATLAS simulation. The geometry and resolution information are extracted from the simulated training samples rather than from a geometric description of the detector.

To maximize the efficiency of the pattern bank for a given size, we order the pattern list by the number of training tracks that match a pattern. We use this number to define the coverage of the single pattern. The larger the number of training tracks passing through the same pattern, the larger is that pattern's coverage. We then fill up the available AM pattern locations using the most frequently hit patterns, the "high-coverage patterns".

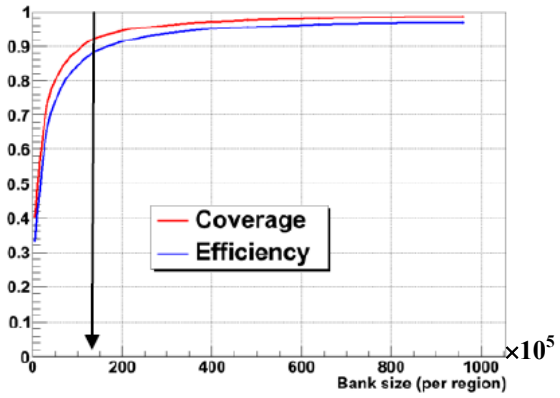

Fig. 1: The coverage and efficiency of the AM bank in the barrel (|η|<1). The SS sizes are 16 strips in the r-φ SCT layers and 22 (36) pixels in the φ (z) direction in the pixel layers. The arrow shows the chosen operating point.

The number of generated patterns depends on the helix parameter range of the training track sample. We have selected ranges appropriate to triggering on electrons, muons, taus, and heavy quark jets. Azimuth, rapidity and z_0 are generated flat over the entire silicon detector. Curvature is generated flat for tracks of $p_T \geq 1$ GeV (chosen for e and μ isolation as well as τ and b daughters), and the impact parameter is flat out to 2 mm in order to be efficient for b-quarks.

Fig. 1 shows a typical curve of track coverage and efficiency versus bank size. The coverage is determined by the bank only, while efficiency is affected by all the FTK algorithms. There is an initial rapid rise as the bank is filled with high-coverage patterns, patterns that tracks commonly match. Beyond that, the curve rises slowly as less probable patterns are added from the tails of the multiple scattering distribution ("low-coverage patterns"). Although the efficiency for real tracks grows slowly in this region, the number of fake matched roads rises nearly linearly with the bank size. Thus we have to choose a size that balances the need for good efficiency with the need to limit the rate of fake matched roads. Moreover, as the luminosity increases, the number of fakes increases rapidly because of the increased silicon occupancy. To counter that, we must reduce the width of our roads to improve resolution.

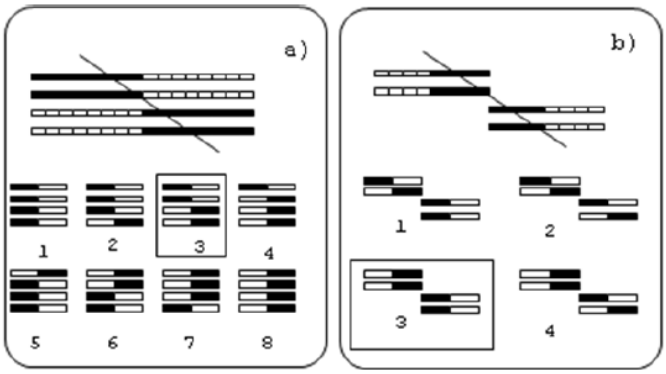

Fig. 2: (a) Eight patterns are compatible with a straight line when each layer is divided into two bins. Pattern number 3 matches the input track. (b) The width of the SS in each layer is further reduced by a factor of two and we test the four possible patterns consistent with the matched pattern in the previous step.

We can use a successive approximation strategy to reduce the pattern width. We could repeat pattern matching on the same event with ever increasing pattern spatial resolution (decreasing pattern width). Coarser resolution is obtained by simply ORing adjacent SSs. Fig. 2 demonstrates how this works for the case of straight line tracks passing through a 4-layer tracker. Fig. 2.a shows the eight possible patterns found when a large road (let's call it "fat road") is divided into two half SSs in each layer. Pattern 3 matches the track's actual trajectory, and it becomes the "parent pattern" for the next stage in which the width of the SSs is further reduced by a factor of two. Figure 2.b shows that there are 4 possible finer resolution patterns, with pattern 3 matching the input track. Since there still is a track candidate, we can further reduce the SS size by another factor of two. This process is iterated until either we reach the final resolution (success) or we are left with no track candidate (a low resolution "fake road" seeded the search for a real track that doesn't exist). A fake road is one that disappears when we use finer resolution.

The pattern bank can thus be arranged in a tree structure (Fig. 3) where increasing depth corresponds to increasing spatial resolution. The tree root corresponds to the incoming fat road. Each node not belonging to the final level represents one "parent pattern" and is linked to its sub-patterns (pattern block), corresponding to a factor of 2 increased SS resolution. We could compare each sub-pattern to the event and every matched pattern is a track candidate that enables the search at the next level. A refined track candidate (thin road) is found

whenever the tree bottom is reached. Reaching the intrinsic detector resolution, the pattern recognition is complete.

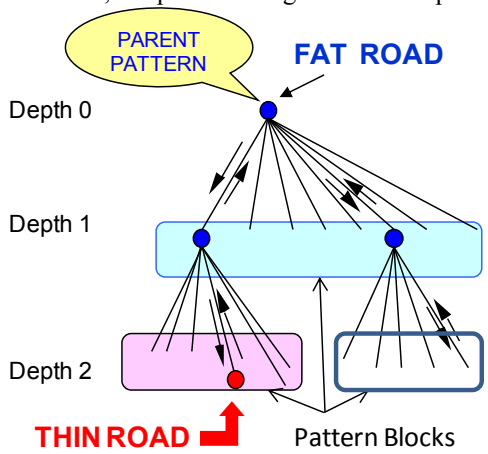

Fig. 3: Hierarchical pattern organization in a tree structure.

However it is not convenient to push the pattern matching search down to the detector intrinsic resolution, since a few remaining hit ambiguities are easily resolved by the track fitter which fits all of the hit combinations in a matched road. The tree search is obviously much faster than a purely sequential search, but it is also much slower than the AM approach; given a certain resolution the AM compares in parallel all the patterns of the corresponding tree level with the event. In the tree search, instead the average number of patterns one has to examine serially to find a single track [6] is proportional to the total number of levels in the tree and to the average number of patterns in a pattern block. If there is more than one track in a fat road, the number of nodes tested during the tree search increases faster than linearly [7] since many fake matches can occur at low resolution.

To maintain the minimum latency for pattern matching, the AM road finding approach is preferable. However, if we narrow the road width using current technology, the needed hardware would become very large and extremely expensive. Fig. 4 shows the coverage and efficiency versus bank size when the SS sizes are divided by two in the φ direction compared to the segmentation of Fig. 1: 8 strips in the r-φ SCT layers and 11 (36) pixels in the φ (z) direction in the pixel layers. The working point of ~120 M patterns in Fig. 4 is roughly a factor 10 larger than that for the bank of Fig. 1 (~12 M patterns). Reducing the pattern width by a factor 2 just in the φ direction requires a bank a factor of 10 larger to provide a similar efficiency.

We propose an elegant solution to this problem, the "variable resolution patterns". It allows the use of a small AM bank, like the one of Fig. 1, while profiting from the positive effects of high pattern resolution (like the bank of Fig. 4).

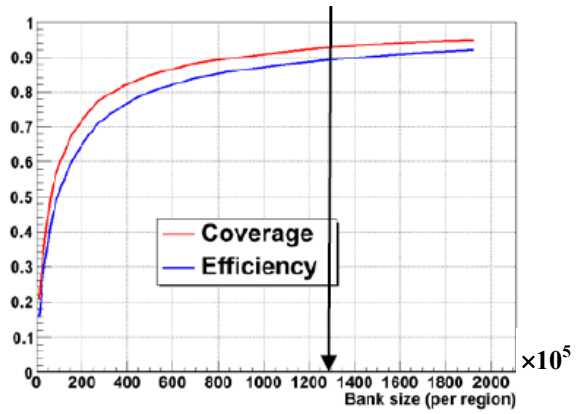

Fig. 4: The coverage and efficiency in the barrel for the bank with pattern width reduced by a factor of 2 along the φ direction. The SS sizes are 8 strips in the r-φ SCT layers and 11 (36) pixels in the φ (z) direction in the pixel layers. The arrow shows the chosen operating point.

III. The Patterns with Variable Resolution

We include in the chip the ability to exploit high resolution for each pattern and each layer, but we use a "don't care" feature (inspired from ternary CAMs) to employ such fine resolution only when necessary. In this way the shape of each pattern[2] can be optimized to improve the acceptance for valid tracks with maximum fake rejection. The goal of the increased resolution is the reduction of the number of fake roads, roads that can be removed just by using a better resolution. Fake roads at the baseline LHC occupancy are a very large fraction of the total number of roads found for an affordable AM system without variable resolution.

High-coverage patterns cause most of the bank size increase when pattern width is decreased. The tree representation of the bank shows that the high-coverage patterns are those that produce many sub-patterns (Fig. 5). High-coverage patterns are very symmetric, being compatible with many possible different tracks as can be seen in the figure.

For this reason the probability that a fake road survives a single step of resolution improvement is high. If the high-coverage pattern matches the event, there is a good probability that more than one sub-pattern will also match the event. To kill high-coverage fake roads, many steps in the tree are necessary. Thus the high-coverage patterns are retained at low resolution in the pattern-bank since the resolution needed to significantly reduce their fakes is beyond today's technological reach.

However the main cause of the fakes found by a standard AM is the large number of low-coverage patterns for a bank with narrow patterns. In Fig. 1 the 90% efficiency is reached easily just after the rapid raising part of the curve. In Fig. 4 instead the high-coverage patterns provide only a limited efficiency (~80%) and the last 10% gain is reached only if a very large number of low-coverage patterns are included in the bank. These low coverage patterns produce most of the fakes,

[2] The shape of a pattern is given by the width of the super strip in each layer.

which can easily be deleted with just one step of finer resolution.

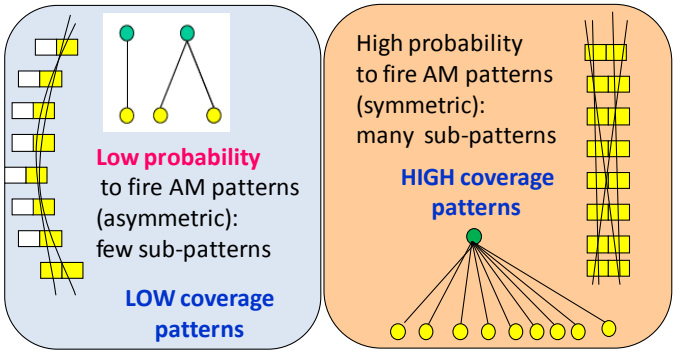

Fig. 5: graphic representation of low-coverage and high-coverage patterns.

For a low-coverage pattern, Fig. 5 shows that, for resolution improved by a factor of two, only a fraction of the half SSs are compatible with any of the sub-patterns. They are the yellow ones. The white ones are not touched by any real track, but they offer all their acceptance to fakes. Going down just a single step in SS width reduces the fake road rate significantly. Simulation with real events confirms this hypothesis (see next section).

The choice of high or low resolution can be separately applied layer by layer. When we divide a SS in two parts, both halves can be touched by real tracks (both halves in the same layer are yellow, using the representation style of Fig. 5) or just one of them is compatible with a real track (one half is yellow and the other one is white). In the first case the layer is used at low resolution applying the "don't care" (DC) feature to the least significant bit of the pattern word during the comparison with the event. In the second case the layer is used at full resolution, reducing by a factor two the area that the SS offers to uncorrelated hits from other tracks. As a result we reduce the number of fake roads, keeping the efficiency high and avoiding the bank size blowing up due to the narrower SS width.

IV. Simulation Results

The simulation program for FTK (FTKSim) processes complete ATLAS events and creates the same list of tracks that will be produced by the FTK hardware. The program serves a number of purposes:

- detailed and reliable evaluation of FTK physics performance by processing complete events produced by the full ATLAS detector simulation,
- evaluation of crucial parameters needed for the hardware design,
- determination of the large set of constants needed for programming FTK, and
- determination of tracking performance parameters for use in fast parametric detector simulation for high statistics studies of physics performance.

These goals are achieved with an intermediate-level simulation that describes the FTK algorithm and internal data accurately, but avoids detailed bit-level simulation of the hardware in order to attain sufficient speed for simulating moderate sized samples of complete events. The core code [8] is based on a similar simulator created for the CDF SVT but with data structures appropriate for ATLAS Athena [9].

The simulated datasets used for performance studies are associated production of a W and a Higgs, with the Higgs forced to decay to either $b\bar{b}$ or $u\bar{u}$.

Samples were produced at three instantaneous luminosities with different number of pile-up events per beam crossing: 17.6, 40 and 75. WH events have been chosen as typical of the multi-jet events that come out of the level-1 trigger. These could be the most complex events FTK has to process.

The table below reports the number of roads found in the event for the different detector occupancies. We use 3 different pattern banks: the low-resolution bank of Fig 1, the high-resolution bank of Fig. 4 and the new "variable resolution" bank. The bank with "variable resolution" using only 1 DC bit has a size intermediate between the low- and high-resolution banks, approximately a factor of 3 smaller than the high-resolution bank. The track efficiency measured on WH events is 90% for muons and 79% for pions with occupancy of 40 pile-up events, which deteriorates by roughly 10% at very high luminosity. The bank coverage (efficiency due to the bank only) is more than 90% for both 40 and 75 pile-up events.

The first two columns of table 1 show that 80-90% of roads coming out of the low-resolution bank are fake roads, disappearing if we reduce the pattern width by a factor of 2. The last column shows that the variable resolution idea reduces fakes very similarly to the high resolution case, with the advantage that the bank size is much smaller.

Table 1: The mean number of roads found in WH events at 3 different luminosities and using 3 different patterns banks: the low-resolution bank of Fig 1, the high-resolution bank of Fig. 4 and the new idea of a variable resolution bank.

Pile-up	Low-resolution	High-resolution	Variable resolution
17.6	645 10^3	135 10^3	123 10^3
40	4.7 10^6	0.73 10^6	0.83 10^6
75	6.8 10^6	0.76 10^6	1.1 10^6

In fact both low-resolution and high-resolution options are not implementable with today's technology, the former because it would require too much hardware to process the roads found by the AM, the second one because it would require too large an AM bank.

Variable resolution offers the solution, allowing a reasonable size for both parts of the system. Variable resolution achieves the same efficiency and same rate of fake roads as the high-resolution case with a factor of 3 smaller bank, thus improving by a factor 3 the effectiveness of the hardware

These results can be further improved using a second step or even third step in resolution using 2 or 3 don't care bits. Preliminary results with 2 don't care bits show a factor of 5 gain in number of patterns for same efficiency and fake performance.

V. The Implementation in the Simulation

As described in Section II and III, the basic idea is to have two different segmentations of the inner detector: a high-resolution (thin) SS segmentation, producing "thin patterns", and a low-resolution (fat) SS segmentation that produces the "fat patterns". The fat SS size is a power of 2 times the thin size (2, 4, 8, …) and the thin SSs are internal partitions of the fat SS.

The simulation starts by producing the thin SS bank, resulting in a very large number of high resolution patterns. Using the geometrical relation between the high resolution and the low resolution SSs, the thin patterns are grouped into families producing the tree structure of Fig. 3. The bank generation is still not complete. The last step is a scan of all the layers used in any fat pattern to set for each layer the status of the DC bits. This can be understood with the help of Fig. 5. If we choose the fat SS size as twice the thin one, each fat SS will be divided into two parts, described by 1 DC bit and two possible scenarios: all the sub-patterns involve the same half SS (in this case the other half SS is not touched by any track, as the white bins in Fig. 5) or they involve both halves (layers in Fig. 5 where no white bin is shown, both are yellow). If both the half SSs are yellow, the DC bit is set to 1, and the lower resolution is used for that layer in the pattern matching simulation. If one of the two half SS is white, the DC bit is set to 0 and the higher resolution is used in the pattern matching. In this latter case we need an additional bit (the position bit, POS) that establishes which half SS is to be used in the comparison with the event.

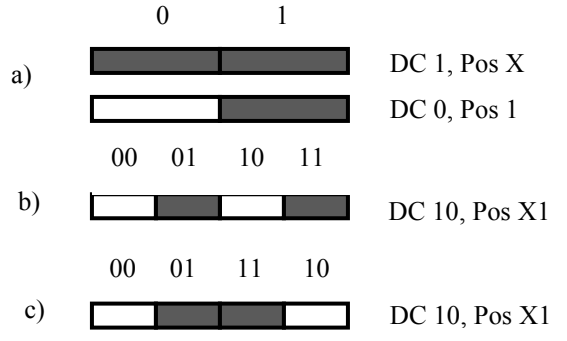

Fig. 6: The active partitions within a fat SS for different DC and position values. The numeric values are showed in binary format.

If multiple DC bits are used, we find that the use of a Grey code improves the ability to group neighboring high resolution SSs. Three examples of the construction of the DC and the position bits are shown in Fig. 6. For each configuration of DC and position bits, the figure shows in dark gray the high-resolution SS that are accepted for a match. In a) there are 2 examples using 1 DC bit; the "X" in the position indicates the value doesn't matter and both positions are matched. In b) there is an example with 2 DC bits, when only the most significant bit is set to don't care (DC=10). With standard binary encoding this configuration accepts two non-contiguous bins. In c) it is seen that using a Grey code provides a more powerful description of the active partition. In the case of DC=10 we can match the two central position. This is not possible with binary encoding.

The bank just described is used in the pattern matching simulation. Two steps are executed: the first step performs the old-style AM chip simulation without taking into account the DC bit information, producing the results of the low-resolution pattern matching (see table 1). After that, the roads found by the first step are tested in more detail: in the layers where the DC bit is not set the hit position is checked at higher resolution. The hits in the SS part selected by the POS bits (yellow bins of Fig. 5) are kept, the others (the one in the white bins) are removed. This procedure rejects all the fake roads that fired during the first step because of hits belonging to the rejected thin, white SSs.

This two step method is not the same used in the hardware but has exactly the same result and helps us debug and understand the rejection flow of the variable resolution patterns.

VI. The Implementation in the AMchip

We have designed a new AMchip prototype with the goal of increasing the pattern density by introducing the DC bits to allow more effective pattern matching. The new device will perform pattern matching with up to 8 layers. Within each pattern the hardware allocates for each layer a CAM word [10] of 15 bits to store the hit position with high resolution. Incoming hits are compared with the stored word to find layer matches. If all or all except one of the layers are matched, the pattern is considered matched and its address is sent out.

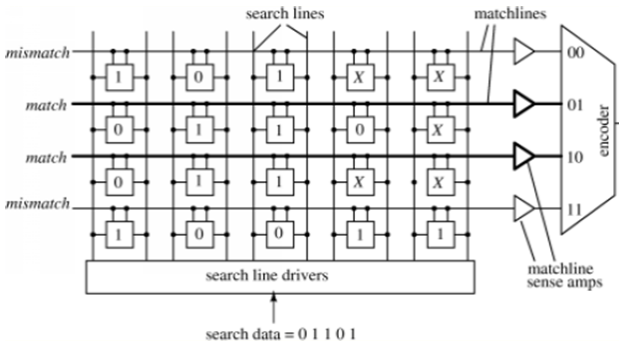

Fig. 7: Logic of 4 CAM words with "don't care" bits.

The 15 bits within each word are configured to store 12 bits plus 3 bits with the don't-care option. By setting one or more of these bits to DC we can allow a single layer to match hits with lower resolution. Since in our application the hit encodes a position, the resolution can be lowered for each pattern and each layer by a factor 2, 4 or 8 depending on the number of DC bits. From the hardware point of view each CAM bit with the don't-care option occupies two times the area of a normal bit. In fact the ternary cell is implemented combining two normal cells. In a don't-care cell three values (0, 1, X) can be stored using two SRAM cells, as showed in Fig. 7, compared to normal cells that only store (0, 1) values. The additional

cost of the don't-care hardware is a 20% increase in the core logic (18 normal cells vs 15 cells). Since the number of patterns needed for same efficiency and same fake rate is reduced by a factor of at least 3, this is a very effective technique.

VII. Conclusions

The variable resolution pattern matching is an innovative idea that makes it possible to reduce the cost, size and complexity of FTK, down to values that allows building it for the expected SLHC luminosities. Using just a single DC bit we gain a factor 3 in the effective number of patterns, with larger gains expected using 2 and 3 DC bits.

Pattern matching techniques are widely used in trigger applications for high energy physics experiments, and most applications can exploit variable resolution patterns to improve the effectiveness of pattern matching.

References

[1] A. Annovi et al., "Hadron Collider Triggers with High-Quality Tracking at Very High Event Rates", *IEEE Trans. Nucl. Sci.*, vol. 51, pp 391, 2004.

[2] J. Adelman et al., "The Silicon Vertex Trigger upgrade at CDF", *Nucl. Intsr. and Meth. in Physics Research A*, vol. 572, Issue 1, pp 361-364, March 2007.

[3] J. Adelman et al., "Real time secondary vertexing at CDF", *Nucl. Intsr. and Meth. in Physics Research A*, vol. 569, pp 111–114, 2006.

[4] J. Adelman et al., "On-line tracking processors at hadron colliders: the SVT experience at CDF II and beyond", *Nucl. Instr. and Meth. in Physics Research A,* vol. 581, pp 473–475, 2007.

[5] A. Annovi at al., "A VLSI Processor for Fast Track Finding Based on Content Addressable Memories", *IEEE Trans. Nucl. Sci.,* vol. 53, pp *24*28, 2006.

[6] M. Dell'Orso and L. Ristori, "A highly parallel algorithm for track finding", Nucl. Intsr. and Meth., A287, (1990) 436-440

[7] M. Dell'Orso, L. Ristori, "VLSI Structure For Track Finding ", Nucl. Instr. and Meth. A 278 (1989) 436.

[8] E. Brubaker et al., "Performance of the Proposed Fast Track Processor for Rare Decays at the ATLAS Experiment.", IEEE Trans. Nucl. Sci. vol. 55, pp 145-150, 2008

[9] The ATLAS Collaboration. ATLAS Computing Technical Design Report, July 2004. ATLAS TDR-017, CERN-LHCC-2005-022.

[10] K. Pagiamtzis and A. Sheikholeslami, "Content-addressable memory (cam) circuits and architectures: a tutorial and survey", Solid-State Circuits, IEEE Journal of, vol. 41, no. 3, pp. 712 – 727, march 2006.