ELSEVIER Nuclear Instruments and Methods in Physics Research A 413 (1998) 367–373

NUCLEAR INSTRUMENTS & METHODS IN PHYSICS RESEARCH Section A

A programmable associative memory for track finding

A. Bardi[a], S. Belforte[a], A. Cerri[b], M. Dell'Orso[c], S. Donati[a], S. Galeotti[a], P. Giannetti[a,*], A. Leger[d], E. Meschi[c], F. Morsani[a], D. Passuello[a], G. Punzi[b], L. Ristori[b], T. Speer[d], F. Spinella[a], X. Wu[d]

[a] *Istituto Nazionale di Fisica Nucl. Pisa, Via Livornese 1291, 56010 S.Piero A Grado (PI), Italy*
[b] *Scuola Normale Superiore, Piazza dei Cavalieri 7, 56100 Pisa, Italy*
[c] *Dipartimento di Fisica, Università di Pisa, Piazza Torricelli 2, 56100 Pisa, Italy*
[d] *Département de Physique Nucléaire et Corpusculaire, Université de Genève, 24 Quai Ernest-Ansermet, CH-1211 Genève, Switzerland*

Received 9 February 1998

Abstract

We present a device, based on the concept of associative memory for pattern recognition, dedicated to on-line track finding in high-energy physics experiments. A large pattern bank, describing all possible tracks, can be organized into Field Programmable Gate Arrays where all patterns are compared in parallel to data coming from the detector during readout. Patterns, recognized among 2^{66} possible combinations, are output in a few 30 MHz clock cycles. Programmability results in a flexible, simple architecture and it allows to keep up smoothly with technology improvements.

PACS: 07.05.Hd; 07.50.-e; 07.50Ek; 42.30.Sy

Keywords: Data acquisition; Electronic circuits; Pattern recognition

1. Introduction

Full custom VLSI technology has been proposed [1] as an efficient solution to solve the pattern recognition problem in high-energy physics experiments, where tracks inside very high multiplicity events must be found in a time span of a few microseconds. A large hadron collider experiment (CDF) has already chosen a chip built along these guidelines, the Associative Memory chip (AM) [2], to implement a fast online tracker for the silicon vertex detector [3].

We present the use of a different technology, Field Programmable Gate Arrays (FPGA), to implement the same algorithm with the following advantages:

- *High degree of flexibility*. The programmability allows fast project reconfiguration to adapt the processor to different detector layouts, allowing easy upgrade and evolution of the system.

* Corresponding author. Tel.: + 39 50 880304; fax: + 39 50 88 0317; e-mail: giannetti@pisa.infn.it.

0168-9002/98/$19.00
PII: S0168-9002(98)00577-4

- *Simple architecture.* The project can be customized to the particular experiment's trigger and DAQ needs. The architecture is simplified by dropping any flexibility that must be built in a custom VLSI. A simple design allows to save chip area and to obtain a more efficient routing. As a consequence high pattern densities, good timing performances and faster prototype development can be achieved.
- *Prompt exploitation of technology advancement.* High-energy physics experiments usually have a very long life. Detector construction requires many years to be completed. Technological evolution is so fast that a dedicated processor can become obsolete in the time between prototype design and production. The use of commercial FPGA devices allows an easier update of the project. Technological improvements can be followed simply by replacing an old device with a pin-compatible new version, which is usually faster and denser. Production can use up-to-date technology.
- *Easy prototype test and debugging.* Programmable chips are 100% tested at the factory. System performances can be studied and predicted with the full device simulator provided by the manufacturer. In-system programmability helps in final trimming and debugging. Boundary scan support allows full test of the printed board connections.[1]

A Programmable Associative Memory (PAM) suitable for CDF has been implemented into the XC5210-5 Xilinx FPGA chip [4].

2. PAM logic

PAM is designed to solve the track finding problem for a detector consisting of a number of *layers*, each layer being segmented into a number of *bins*. When charged particles cross the detector they hit one bin per layer. Each track generates a set of *hits*.

The coordinate of each hit will actually be the result of some computation performed on raw data, for example the center of gravity of a cluster. These calculations have to be done upstream and only the resulting *binned* coordinates have to be transmitted to our device.

For each event a number of tracks traverse the detector and a particular configuration of hits is generated: we use the word *event* to indicate this hit configuration. We know the bins that have been hit and from this information we can reconstruct the trajectories of all the particles.

To this end we list all possible tracks that can go through the detector. A *hit pattern*, consisting of one hit bin per layer, corresponds to each potential track. All the different hit patterns are stored in a sufficiently large memory, that is called the *pattern bank*.

For each event we scan the pattern bank and compare each pattern to the event that has occurred. A track candidate is found only when all the hits in the pattern are present in the event.

PAM stores the pattern bank: each memory cell is large enough to hold one pattern and has enough logic built in, to compare its content to the event.

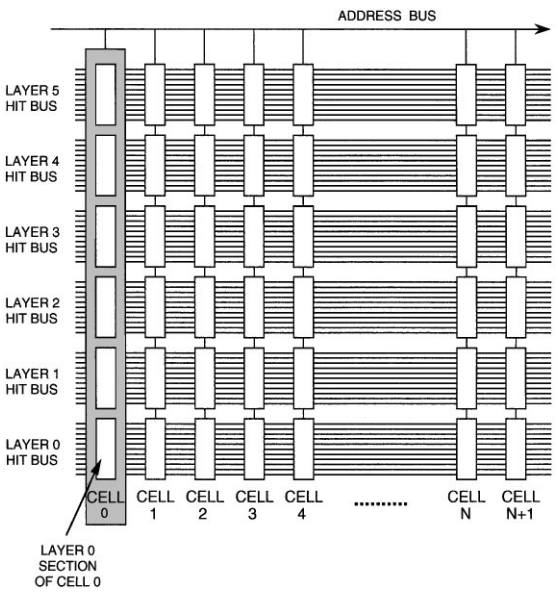

Fig. 1. General layout of an associative memory for storing the pattern bank.

[1] Last generation FPGAs usually support all the boundary-scan instructions as specified in the IEEE standard 1149.1.

A possible layout for this device is sketched in Fig. 1. Each column represents one memory cell and is structured into a number of sections, one section per detector layer (six of them are shown). Each section stores the coordinate of one bin on the corresponding layer. The sections in a memory cell define a pattern by specifying one bin per layer. The Hit Buses connect together all sections of the different cells corresponding to the same layer. For every event the coordinates of all the hits in each layer are sequentially transmitted on the corresponding Hit Bus (INPUT). In a slightly different architecture, hits of different layers may be multiplexed also on a single Hit Bus.

Every section continuously compares its content to what is on the bus and, if a match is found in all sections of the same cell, the pattern is a track candidate. Each pattern is identified by a serial number, called the *address*, in the pattern bank. The addresses of all track candidates are transmitted sequentially on the Address Bus (OUTPUT).

3. Guidelines for the PAM design

The experience gained with the full custom AM [2] sets the main guidelines to design a new device. In the process of fitting the device for CDF we traded complexity of noncritical parts with benefits of other critical functions:

1. An operation code selects one out of several operations of the AM. This allows its use in different experiments with very different system environments. A dramatic simplification characterizes the PAM: it automatically switches from one operating condition to another with no need of any specific codes. In particular, one of the operating conditions of the AM is the pattern bank downloading. This is done before the experiment begins to take data, and is therefore not a time critical function. To download the patterns in the device, PAM allows the use of a dedicated hardware built by the factory of the FPGA chip for its programmability.
2. The AM uses two 30 MHz clocks that have to be distributed to all chips within the system maintaining the capability of controlling the relative delay of the four clock edges with a precision of a few nanoseconds. This is necessary to handle the capability to swap to any operating condition at given time. The simplified FPGA project does not allow to change the operation flow while being run. Any change can be done only by modifying the design and downloading a different logic scheme into the chips. This simplification allows the distribution of a single 30 MHz clock (CLK); moreover only the raising edge transition is used by the PAM.
3. The associative memory is expected to receive large amounts of data from the detector and output a much smaller number of the candidate tracks, providing a very strong information reduction. For this reason the time critical function is the detector data write-in to make a comparison with the pattern bank (INPUT). The readout of the candidate tracks (OUTPUT) is much less critical. The CLK frequency is internally divided by a programmable factor to generate a lower-frequency clock, so that the PAM output rate is lower than 30 MHz. A slower output rate allows to dedicate the maximum placement and routing efforts to the input logic working at the CLK speed. A lower output rate of the single device is not a disadvantage to the whole system, since many PAMs are read out in parallel and the 30 MHz output rate is restored where many slower streams are merged into a single faster one (see Section 6). Next generation FPGA devices will have an internal Phase-Locked Loop (PLL). In this case the low-frequency clock can be distributed on the board and the high-frequency one can be generated inside the chip.
4. An interesting advantage is the capability to overlap the INPUT and OUTPUT operations. Candidate tracks can be read out immediately on being found. This means that a few clock cycles after the last detector hit has been distributed to the PAMs, the last candidate track will be read out. Pattern recognition is fully performed in parallel with the detector readout. The AM cannot overlap the INPUT and OUTPUT functions because of its higher complexity. Track readout begins when the INPUT is completed.

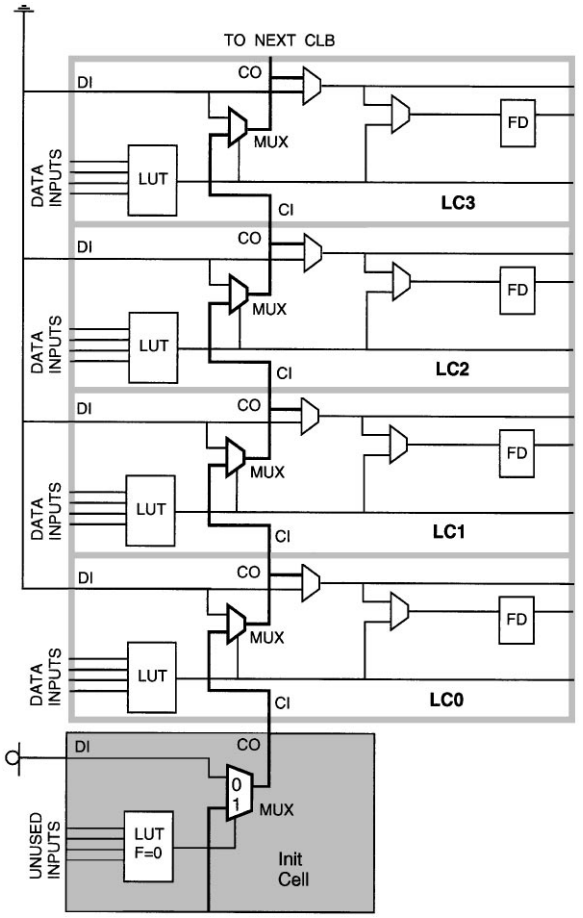

Fig. 2. The XC5200 Configurable Logic Block. The Init Cell, used to initialize the carry chain, belongs to the adjacent CLB.

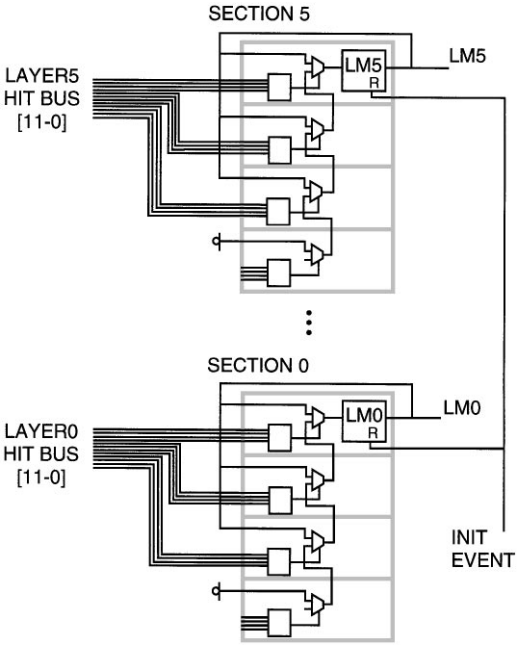

Fig. 3. An example of *Pattern Decoding Logic* implemented in a XC5200.

4. PAM implementation

The structure of the XC5200 device family [4] is suited to implement in the same device many parallel, low-cost, fast, and wide pattern decoding functions.

These devices are composed of many elementary Configurable Logic Blocks (CLBs). We briefly describe the CLB shown in Fig. 2. Each CLB consists of four Logic Cells (LC0, … , LC3). Each Logic Cell has a 4-input 1-output LookUp Table (LUT) followed by a D-type Flip–flop (FD). The high-speed carry logic (highlighted lines and MUX multiplexers in Fig. 2) is used for high-speed wide pattern decoding. The carry chain, initialized to a high level by the Init Cell, is propagated only if the LUT output is high. If the LUT output is low, a low level logic is selected through the direct input (DI). The carry chain is also propagated between adjacent CLBs through dedicated fast connections shown on the top and bottom of the CLB in Fig. 2.

Fig. 3 shows how a single memory cell can be implemented in the XC5200 architecture (Pattern Decoding Logic). Two out of the six sections are shown.

During the INPUT operation, the event hits are sent on 6 Hit Buses, one Hit Bus per detector layer. Each LUT monitors a 4-bit field of the Hit Bus and outputs a high level only if a match is found. Matches of different bit fields are logically ANDed by the carry chain. If all hit bits match, the Layer Match flip–flop (LM) is then set. The pattern is stored in the memory cell at power on, when the configuration bitstream is downloaded in the chip. A hit is encoded by storing in each LUT a logic one in the location addressed by the four hit bits routed to that LUT.

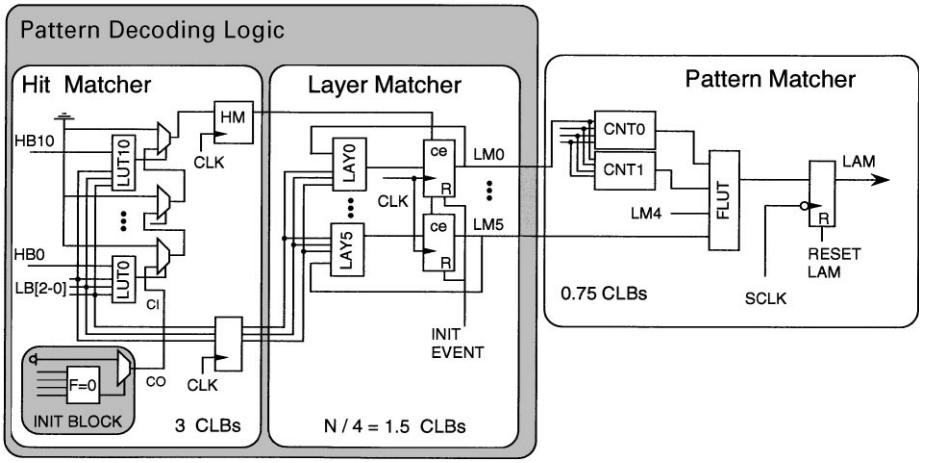

Fig. 4. The logic of a high-density memory cell. N = 6 is the number of detector layers.

Any memory cell that has all the LM flip–flops set is a track candidate because all the hits that define that pattern are present in the event.

Implementation in Fig. 3 is a simple example of the Pattern Decoding Logic. A more efficient placement can be obtained in the XC5200 architecture when different layers are not read out in parallel and the hits of different layers are serialized into a single stream. PAM is designed specifically for this case.

The match logic for a single memory cell is shown in Fig. 4 for a 6 layer detector (or subdetector), with 2^{11} bins per layer (11 bits encode the hit coordinates), amounting to a total of 2^{66} possible hit patterns. This configuration is what is needed in the CDF and is implemented in the AM chip.

The Pattern Decoding Logic is now divided into two segments. The comparison function is implemented in the first segment, the *Hit Matcher*. Since the same carry chain is used for all the layers in the memory cell, as described in detail below, a single flip–flop is not sufficient to memorize the matches of all six layers. For this goal a second segment, the *Layer Matcher*, has been added. The total number of *layer matches* is monitored by the *Pattern Matcher*.

In more details (Fig. 4):

- The *Hit Matcher* uses one 11-bit Hit Bus (HB) and a set of eleven 4-input LUTs, one LUT per hit bit, with outputs connected in a single carry chain. All layers share the same LUTs. Hits of different layers are multiplexed on the Hit Bus. A 3-bit Layer Bus (LB) encodes the layer number of the hit on the Hit Bus. The same configuration allows up to 8 layers: 6 layers were used in the actual implementation. For each LUT, three input bits come from the Layer Bus and are common to all LUTs. The fourth input bit, specific of the LUT, is one of the 11 HB bits.
The 2^4 1-bit locations of each LUT are organized as 8 2-location tables, one table per layer. The layer table is selected by the Layer Bus. The two locations of the same layer table are addressed by the HB bit.
At initialization the hit pattern is memorized in the Hit Matcher LUTs. The hit coordinate of each layer is stored in the 11 layer tables, one bit per table, according to the following redundant code. The high address location contains the value of the hit bit and the low address location contains its complement, as shown in Table 1.
During INPUT the event is compared to the pattern by sequentially presenting the hits on the Hit Bus and the hit layers on the Layer Bus. Each HB bit under test condition matches the stored hit bit when it addresses the table location that contains a logic 1. The Hit Match flip–flop

Table 1
Redundant code for storing one hit bit in a 2-location layer table

Hit bit	Table location 1	Table location 0
0	0	1
1	1	0

(HM) is set when all the HB bits match the stored hit.

Within the limit of 8 layers, the used chip area is independent of the number of layers. The timing performances are strongly dependent on the chip layout. The structured, hierarchical routing resources of the chip [4] are efficiently used to give high priority to this most critical circuit sections. As a result of the optimization the maximum input rate is 30 MHz.

- The *Layer Matcher* is a layer decoder driving a single flip–flop per layer. The flip–flop of each layer is set when the Hit Matcher signals that the corresponding pattern bin has been hit (Layer Match). Feedbacks memorize the Layer Matches.
- The *Pattern Matcher* monitors the Layer Matches and identifies the Pattern Match. Allowing detector inefficiencies, the pattern can still match when the number of matching layers exceeds a given threshold. Counting and comparison are enforced by the lookup tables CNT0, CNT1, and FLUT. Memory of the Pattern Match is kept in the Look At Me flip–flop (LAM) until the pattern is read out by the Readout Logic. As soon as the pattern is read out the LAM is reset and maintained in this state until the event processing ends. The LAM Slow CLocK (SCLK) is slower than the system clock CLK. As described in the introduction, the CLK frequency is internally divided by a programmable factor to generate SCLK. The PAM output rate is then lower than 30 MHz.

Fig. 4 also shows the chip area used by each segment of a memory cell in units of CLBs (to be compared with the chip size of 324 CLBs). In summary, for an N-layer detector of 2^{11} bins per layer, the simple configuration in Fig. 3 fills $N + 0.75$ CLBs, while the compact configuration in Fig. 4 uses $N/4 + 3.75$ CLBs.

As described in Section 2, each pattern is uniquely identified by its address in the pattern bank. The patterns inside each chip are ordered and numbered. The chips on a board and the boards in the whole system are ordered and numbered as well. The unique pattern address in the whole pattern bank is built by concatenating the pattern number in the chip, the chip number in the board and the board number in the system.

The Readout Logic in each PAM has to output the internal address of a matched pattern when requested from outside. The protocol is the same of the AM [1]: the Output Ready signals the presence of at least one track candidate ready to be output. The input signal Select directs PAM to output next track candidate on the Address Bus of Fig. 1.

The use of LUTs, 3-state buffers, and carry logic inside the chip allows a compact Readout Logic. To exploit efficiently the 4-input LUTs, the memory cells are organized into groups of four cells, *Quartets*. Priority encoders control the memory cells inside each Quartet and the Quartets inside the chip. The grouping of patterns into Quartets is done in such a way that the Readout Logic is located very close to each memory cell in the Quartet. The Readout Logic occupies less than 2.5 CLBs per Quartet.

5. Packing patterns

The pattern bank is a large collection of sets of coordinates (Fig. 1): many bit fields of these coordinates are repeated a number of times in different memory cells. Thus, PAM compares the same bit fields in different locations a number of times. If we could reduce this redundancy, decoding a shared bit field once per chip and using it whenever needed, large amounts of space can be saved. This means that there are *easy to pack* subsets of patterns that need less information to be stored than the *generic* patterns. The area reduction is paid with a more complex software for organizing patterns into classes to be downloaded in the PAMs.

The PAM pattern density depends on the kind of reduction applied and on the choice of the particular FPGA device. The present design places 48 generic patterns into a XC5210-5TQ144 device for a detector (or subdetector) not more complex than 6 layers and 2^{11} bins per layer. With a mix of the *generic* and the *easy to pack* patterns, 64 patterns can be easily placed within the same chip.

Inside the chip the number of layers can be traded for the number of bins per layer: the device can fit up to 8 layers if the number of bins is lessened and vice versa it can fit more bins per layer if the number of layers is lessened.

6. Putting chips together

A number of chips can be connected to implement a larger Programmable Associative Memory. Some logic is needed to expand the memory.

The GLUE chip is a device that controls the output of a PAM array. It is an evolution of the corresponding device described in Ref. [1]. The GLUE is an internally pipelined multiplexer from 8 to 1 controlled by a priority encoder. Up to 8 pattern addresses can be read in parallel from as many Address Buses and merged inside the GLUE into a single 30 MHz output stream. Each Address Bus is driven by several PAMs.

On the contrary, the Hit Bus is distributed to the PAM array through internally pipelined fanout chips called INput DIstributors (INDI).

A compact PAM unit containing 64 PAMs on a VME board (32 chips per side) read by a single GLUE and fed by two INDIs is under development. The capability of moving signals to different pins with few chip constraints helps in realizing a good layout and a good routing of the PAM unit, with only 4 mm of interchip distance. The physical size of a PAM unit is $4.4'' \times 11''$.

Larger pattern banks can be achieved by placing more units on the same board or using more powerful chips. In the former case a second level of multiplexing is necessary. The GLUE of each PAM unit is connected to the Head GLUE (HGLUE). The HGLUE receives the Address Buses from the GLUEs and merges them into a single output stream. Both the HGLUE and GLUE maximum output rates are 30 MHz.

References

[1] M. Dell'Orso, L. Ristori, Nucl. Instr. and Meth. A 278 (1989) 436.
[2] S.R. Amendolia et al., IEEE Trans. Nucl. Sci. 39 (1992) 795.
[3] S. Belforte et al., IEEE Trans. Nucl. Sci. 42 (1995) 860.
[4] XILINX, The Programmable Logic Data Book, XILINX The Programmable Logic Company, 2100 Logic Drive, San Jose, CA 95124-3400.