A Low-Power Pipelined CAM for High-Performance IP Routing

Pedro Echeverría, José L. Ayala, Marisa López-Vallejo
Departamento de Ingeniería Electrónica
Universidad Politécnica de Madrid (Spain)
Email: {petxebe,jayala,marisa}@die.upm.es

***Abstract*— Packet switched networks such as the Internet require efficient IP routing in order to manage the traffic flows. In these environments CAM memories play a key role because they provide the address resolution time. This paper presents a practical implementation of a low-power CAM oriented to high-performance IP routing. The architecture devised shows optimal results in terms of area, speed and power consumption for these search-based applications by proposing a pipelined implementation split into banks with a reduction of the parameter word. Experimental results show how the proposed architecture provides significant improvements in terms of power and speed.**

I. Introduction

Content-Addressable Memories (CAM), especially fully parallel CAM, provide an exclusive fast data-search function by accessing data by its content rather than its memory location. Nowadays, it can be found a wide range of applications taking advantage of the CAM function: lookup tables, databases, associative computing, data compression, etc. Recently in the network computing era, fast lookup tables are required for address resolution in network switches and routers such as LAN bridges/switches, ATM switches, and layer-3 switches. Moreover, CAM's fast search functions are especially useful in supporting the quality of service (QoS) required for real-time applications like multimedia data transmission. Even faster search operations are desired for higher speed communications networks like OC-192 and OC-768 where address resolution within less than 10 ns is required.

CAMs are fully associative storage devices that store fixed-length binary words in any location. The memory can be queried to determine if a particular word, or key, is stored, and if so, the address at which it is stored. This search operation is performed in a single clock cycle by a parallel bitwise comparison of the key against all stored words. Since it is possible for multiple CAM entries to match the same query, CAMs often include priority resolution logic to return a single match. This is often a simple mechanism that returns the first match in the device, requiring the set of stored words to be sorted according to priority.

Obviously, the quick search access of the CAMs has a clear disadvantage, their low memory density, mainly due to their area-consuming memory cells and the difficulty of implementing the column address. Another limiting factor to be considered is that power consumption in CAM is still high in comparison with RAM of similar capacity. The main reason for this is the large current and large power consumption in the search operation due to the inherent nature of CAM's parallel search. In this operation, the input data which has been searched for is sent to all memory locations, activating all their array cells for simultaneous data comparison. Therefore, current flows in major data system circuits including long heavy data lines and bitlines for all cell arrays. Another main power consumers during the search operation are the match detection circuits (one for each data word) that include heavy word match lines, and other search-related circuits such as encoders needed for the selection of only one valid match. From this combination of the parallel search power consuming factor with the extra hardware cost needed for its implementation comes the difficulty to achieve high density integration in CAM memories.

Packet switched networks such as the Internet require efficient IP routing in order to manage the traffic flows. The relentless increase in link speeds and traffic volume imposes astringent constraints on IP routing solutions. This paper presents a practical implementation of a low-power CAM oriented to high-performance IP routing. The architecture devised shows optimal results in terms of area, speed and power consumption for these search-based applications by proposing a pipelined implementation of a precomputation-based CAM (in the following PB-CAM) split into banks with a reduced parameter word. On the one hand, the use of a pipeline structure significantly improves the access time. On the other hand, the banked implementation and the modifications of the parameter extraction circuits provide a very low power operation in the CAM.

The paper structure is as follows. Next, an overview of the CAM architecture and operation is summarized in section II. The previous works on this area are reported in section III, and the proposed architecture is deeply described in sections IV and V. Finally, some conclusions are drawn.

II. CAM Overview

Figure 1 shows a block diagram of a simplified conventional (non-pipelined non-hierarchical) CAM architecture. The CAM has four horizontal words (CAM entries) and four bits per entry. The CAM compares the searched data (i.e., 1001 in the figure) to all the entries in the CAM, and identifies the words that match. Whenever a search operation happens, the vertical search-lines (SLs) are reset to ground and the horizontal

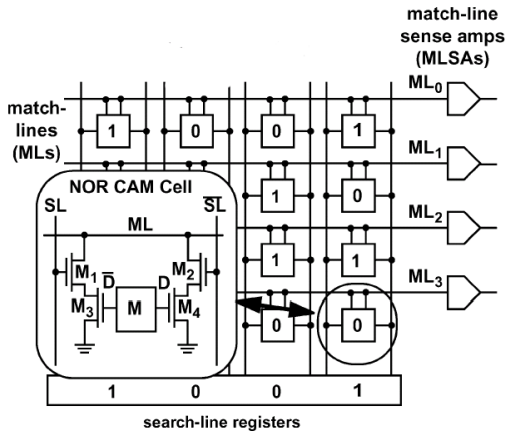

Fig. 1. Simplified CAM architecture [1]

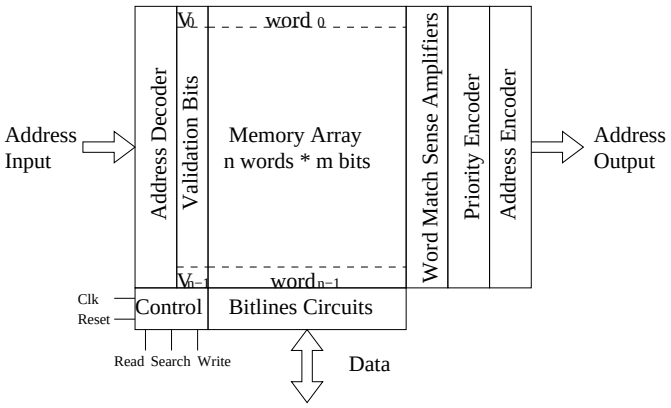

Fig. 2. General scheme for the the CAM architecture.

match-lines (MLs) to V_{DD}. The precharge of the MLs to V_{DD} puts them all temporarily in the match state while the discharge of the SLs prevents wasting of direct-path current during the subsequent precharge of the MLs. When the search operation is complete, the match-lines that remain in the match state will identify the words that match the search data.

After the ML precharge, the search-line registers drive the search data onto the differential search-lines ($\bar{SL}$). Then each CAM cell compares its stored bit against the corresponding search bit on the SLs. The inset of Figure 1 illustrates the schematic of a NOR-based CAM cell. This cell consists of a memory cell, M, which is an SRAM cell in this work [1], and four compare transistors arranged in two pulldown paths between ML and ground. In the cell, a mismatch (or miss for short) between SL and D results in a series path from the ML to ground. On the other hand, a match between SL and D results in no path from ML to ground. The pulldown paths of the individual CAM cells combine on the ML like a dynamic NOR to form either a path to ground (in the case of any miss in the word) or no path to ground (in the case of a full match). In other words, any single miss in any of the cells of a word creates a path to ground that discharges the ML (indicating a miss). Conversely, if all bits of a word match, then the ML remains precharged high (indicating a match). In the example of Figure 1, the search data, 1001, matches the uppermost word in the array. Hence, its associated ML (ML_0) remains high indicating a match, while all the other match-lines discharge to ground, indicating misses. The match-line sense amplifiers (MLSAs) are used to distinguish a match from a miss, often using a threshold voltage as the reference.

As mentioned earlier, the two main sources of power consumption are the highly capacitive MLs and SLs.

III. Related Work

Content addressable memory (CAM) is widely considered as the most efficient architecture for pattern matching required by the LZ77 compression process. In [2], a low-power CAM-based LZ77 data compressor is proposed. By shutting down the power for unnecessary comparisons between the CAM words and the input symbol, the proposed CAM architecture consumes less power than the conventional implementation without noticeable performance penalty.

Based on memory traces, which usually cause tag mismatch within the lower four bits, in [3] a new serial CAM organization is proposed which consumes just 45% more than a single tag RAM read and is only 25% slower than the conventional, parallel CAM. In [4] this work is extended to exploit the address patterns commonly found in application programs, where testing the four least significant bits of the tag is sufficient to determine over 90% of the tag mismatches.

In [6], the proposed CAM word structure adopts a static pseudo nMOS circuit that not only improves system reliability, but also prevents using clock signal to drive the overall system. In order to reduce static power occurred in the proposed CAM word structure, a precomputation approach is used to turn off most pseudo nMOS circuits. This approach is extended in [5] with a design based on a precomputation skill that saves not only power consumption of the PB-CAM (Precomputation-Based CAM) system, but also reduces transistor count and operating voltage of the PB-CAM cell.

The work described in [7] derives power models for four low-power CAMs from the fCV^2 base model. Attending to this work, CAM has three major power-sinking sources: evaluation power, input transition power and clocking power.

Also, [8] presents a new CAM cell with a pMOS match-line driver which reduces search rush current and power consumption, allowing a NOR-type match-line structure suitable for high-speed search operations.

Despite the previous approaches to describe a low-power implementation of a CAM, these works have not considered the system operation of CAMs, with their hard constraints in terms of access time an power. The work proposed here introduces a pipelined implementation of CAM memories targeting both low power and high performance. Next sections will describe the main improvements proposed in this paper with respect to the traditional and PB-CAM implementation of the CAM.

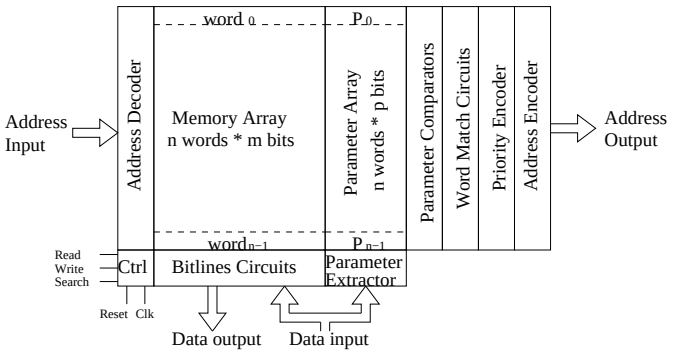

Fig. 3. General scheme for the the PB-CAM architecture.

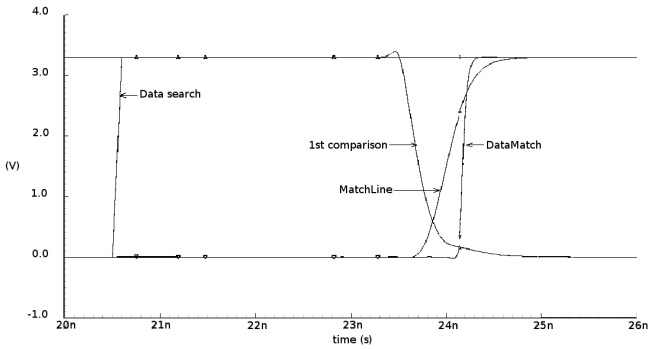

Fig. 4. Functional simulation of the comparison processes (*match*).

IV. Proposed Architecture

A. General Organization

A general CAM architecture usually consists of the memory array with a valid bit field, the address decoder, the bitline precharger, the word match circuit, and the address priority encoder (see Figure 2). The valid bit field indicates the availability of stored data. During the data search operation, the input data is sent into the CAM to be compared simultaneously with all those valid data stored in the CAM. If the data is found, an address from among those matches of comparison is sent to the output.

To minimize the power consumed during the comparison, one of the best approaches is to reduce the comparison operations to a minimum. This can be carried out by a scheme based on the precomputation of a simple parameter which, preferably in a unique way, characterizes every data word [6]. An example of this kind of parameters may be the number of ones in the word. In our work the CAM architecture adopts the ones count function to perform the parameter extraction, because the ones count function filters a large amount of unmatched data with a small bit length.

The memory organization (depicted in Figure 3) of the proposed PB-CAM architecture is composed of the memory cell array, the parameter memory, and the parameter extractor. This parameter extractor has been implemented with a chain of full-adders to perform the ones-count function.

During the data writing operation, the parameter extractor calculates the parameter of the input data, and then stores the input data and its parameter into the data cell array and the parameter cell array, respectively.

In the data searching operation, in order to reduce the large amount of comparison operations, the operation is separated into two comparison processes. During the first stage, the parameter extractor calculates the parameter of the input data and performs a preliminary comparison (the parameter comparison circuits compare in parallel this parameter of the input data with all the parameters stored in the parameter memory). Based on the two comparison processes, since there are a majority of mismatches between the stored parameters and those belonging to the input data, the number of comparisons in the second comparison process is largely reduced. Figure 4 shows the functional simulation results for a *match* in the two comparison processes. The parameter comparison process is also known as the *precomputation process*.

To complete this CAM-search engine for an IP routing application an associated RAM is needed. In the CAM memory, the tags searched will be stored (the input addresses). In the RAM memory, the output addresses associated to each tag will be found. Therefore, the result of a search operation in the CAM memory is directly mapped into the RAM memory. In this way, the address encoders of Figures 2 and 3 can be removed, using the Priority Encoder result as the RAM controls. This implementation can be clearly seen in figure 8.

B. Pipelined Architecture

A clear way to improve the access time of a CAM is the use of a pipeline structure, which additionally promises greater scalability in the performance and density of IP routing processors. In our approach the devised pipeline configuration includes the following operations:

- **READ operation:** EXT - SEARCH - READ_R
- **OVERWRITE operation:** EXT - SEARCH - WRITE_R
- **WRITE operation:** EXT - DEC - WRITE_CR

where READ is the read operation in the RAM memory after the tag is found in the CAM, OVERWRITE is the search and write operation on the RAM memory, and WRITE is the operation to write a tag and its data in both CAM and RAM memories. The pipeline stages defined within those operations are EXT (parameter extraction), SEARCH (data comparison in the CAM), DEC (decodification of internal address, common for both RAM and CAM), READ and WRITE.

However, this three stages pipeline shows a structural and data hazard, as depicted in figure 5.a [1] . This hazard is produced in the CAM structure between the READ (or OVERWRITE) operation and the WRITE operation because the CAM area is simultaneously accessed by the second and third stages, respectively. This problem can be solved by including a fourth pipeline stage splitting the WRITE operation into WRITE_C and WRITE_R (see figure 5.b). All the CAM

[1]Resources (parameter extractor, address decoder, CAM and RAM memories) and data are shown in the plot.

Instruction	1	2	3	4	5	6
Read	P.Ext	C.Mem	R.Mem			
Write		P.Ext	A.Dec	R.Mem C.Mem		
Overwrite			P.Ext	C.Mem	R.Mem	

a) 3 Stage Pipeline

Instruction	1	2	3	4	5	6
Read	P.Ext	NOP	C.Mem	R.Mem		
Write		P.Ext	A.Dec	C.Mem	R.Mem	
Overwrite			P.Ext	NOP	C.Mem	R.Mem

b) 4 Stage Pipeline

Fig. 5. Structure of the proposed pipeline.

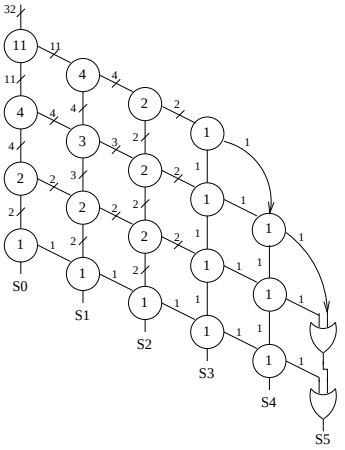

Fig. 6. Parameter extractor (traditional implementation).

accesses are in the third stage and the RAM accesses in the fourth.

- **READ:** EXT - NOP - SEARCH - READ_R
- **OVERWRITE:** EXT - NOP - SEARCH - WRITE_R
- **WRITE:** EXT - DEC - WRITE_C- WRITE_R

The second stage now means a stall in the Read and Overwrite operations but can also be used for further low-power capabilities, as they are the use of voltage scaling techniques (see [9]).

C. Parameter Extractor

The parameter extractor implements a one's count function in order to perform the tag comparison. The implementation of this module uses several area and power-demanding full-adder circuits. In Figure 6 these full-adders are represented with circles (including the number of FA units inside), while the arcs show the number of lines that are propagated to the full-adders in the same level (carry) or following level in the matrix of circuits.

This implementation has been improved as Figure 7 shows. The inclusion of controlled delays in the system allows us to remove several full-adders and exchange others with half-adders (those with two single lines as inputs) which show

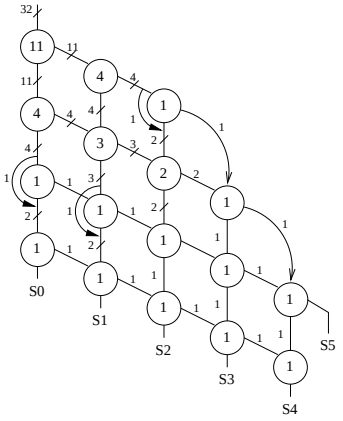

Fig. 7. Parameter extractor (improved implementation).

better behavior in terms of area, power consumption and speed.

As can be observed a simplified implementation of the parameter extractor has been obtained, with very reduced number of transistors and the corresponding optimization on power dissipation and delay.

V. Banked Implementation

The CAM architecture presented in previous sections reduces the power consumption of the original CAM implementation while still meeting the performance constrains. However, the total power consumption of the logic can still be too high for low-power applications. This section introduces a further improvement of the architecture to increase the power savings obtained.

This technique employs the pre-computed parameter to perform a power-aware ordering of the data. The parameter extracted allows us the classification of the memory contents based on the one's count. This classification can be used to store the memory contents in such an efficient way that the search operation is restricted to a smaller memory size.

A. Main Idea

The order of the memory data attending to the one's count parameter allows to split the memory architecture into independent banks. Every data in the bank has the same value for one subset of the parameter (ex. *N-least significant bits*). The logic needed for this ordering is very simple and does not present a serious overhead in terms of delay and energy consumption. Moreover, there are several advantages of using a banked implementation (see Table I).

Due to the banked implementation of the memory, the operation of the architecture is restricted to just one bank every cycle. One of the advantages of this banked structure is the reduction of the dynamic power consumption due to the reduced charge in the bit lines (the driven line is simplified to the bit line of just one bank of the memory). This behavior is also shown by the parameter lines and also has a positive influence in the memory speed.

The complexity of the logic shared for the banks (buffers, priority encoders and address decoders) is reduced when the

TABLE I

ADVANTAGES OF BANKED IMPLEMENTATION.

	Bank structure	Parameter reduction
Area	Common circuitry simplification	Parameter size reduction NAND logic simplification
Power	Reduced # of comparisons Reduced line charge Common circuitry simplification	Reduced # of comparisons Reduced line charge NAND logic simplification
Delay	Common circuitry simplification Reduced line charge	NAND logic simplification

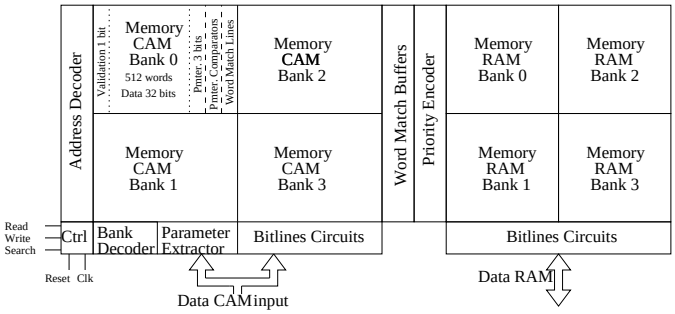

Fig. 8. Banked architecture (4 banks implementation).

bank approach is applied. This simplification saves area, power consumption and improves the delay of these devices.

B. Reduction of Parameter Size

Every word stored in the memory bank has the same value for one subset of the parameter (the one used to select the bank). Therefore, this subset does not need to be stored in the memory and the parameter size is reduced as well as the number of comparisons. For example, in the implementation shown in figure 8, the memory array has been split into four banks, and therefore two bits of the parameter can be omitted.

This simplification allows us to reduce the number of memory cells required and, in this way, the area of the memory. Moreover, the power consumption related to the parameter comparison is reduced since the line charge is smaller. Finally, the logic needed to perform the parameter comparison is significantly simplified.

Finally, it is needed to remember that the parameter comparison in a PB-CAM is divided into two parallel comparisons (filtering and data). The filtering comparison is also affected positively by the banked implementation. When the working bank is decided, the parameter is partially compared because only those words in the selected bank could give a positive result in the search operation. As a result, the number of filtering comparisons is divided by the number of banks.

VI. EXPERIMENTAL WORK

The experimental work and evaluation of the previous architectures has been carried out with Spice simulations in the Cadence environment. We have used the $.35\mu m$ technology from Austria MicroSystems, with dual-poly quadruple-metal CMOS process.

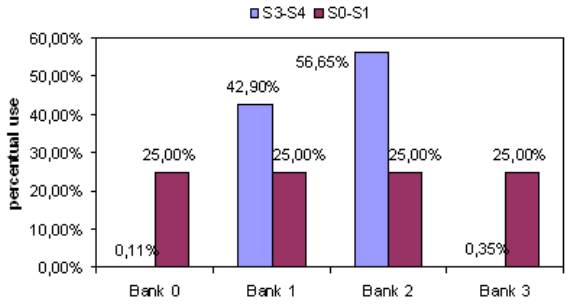

Fig. 9. Percentual use of (S0-S1) and (S3-S4)

A. Baseline Architecture

The baseline architecture has been implemented with the required specifications to execute an internet table routing application. This memory presents a fully-parallel architecture with 32 bits/word CAM and 32 bits/word RAM.

The memory capacity is 2048 positions split into 4 banks and a single 512-positions priority encoder shared by all banks. To select among the 4 banks, 2 out of the 5-bit parameter word have to be selected. Among different options, we have evaluated to use S0-S1 or S3-S4 to select the working bank.

To preserve the spacial locality in the input data, the bits S3-S4 could be selected. However, the optimal design of the banked architecture claims for a homogeneous use of every bank, and the bits have to be carefully selected.

Figure 9 shows the percentual use of the four banks when S0-S1 and S3-S4 are used to select the working bank. For these simulations, the 2^{32} input data have been supposed to have the same probability. As can be seen, the selection of S0-S1 shows better behavior for this application since the four banks are homogeneously used.

With this selection, every data word in the baseline architecture will be composed of the validity bit, three bits for the parameter (S2-S3-S4) and the 32-bits data.

B. Experimental Results

Based on the proposed pipelined PB-CAM word structure and cell circuit design, the PB-CAM architecture achieves low-power, low-cost, and high-performance features. The measured search access energy consumption for our implementation is 31.5fJ/bit/search, which compared with the 86 fJ/bit/search of the baseline architecture [5] represents a 63.3% improvement. Also, the obtained 3 ns latency represents a 70% improvement with respect to the baseline architecture, and a 57% with respect to our non-pipelined implementation.

As was said, the most power consuming task in the CAM access is the search operation. An analysis of the number of memory cells excited during the search operation has been performed. This evaluation gives an overview of the behavior in terms of power consumption for several memory capacities and word lengths. These results are shown in Figures 10 and 11 where the comparison with the tradition implementation, PB-CAM and banked PB-CAM with pipeline can be observed.

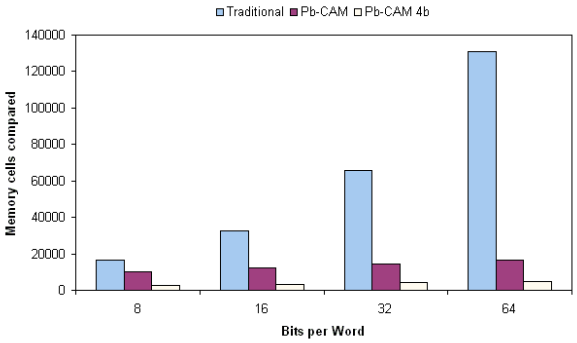

Fig. 10. Activated cells for a fixed memory capacity (2048b)

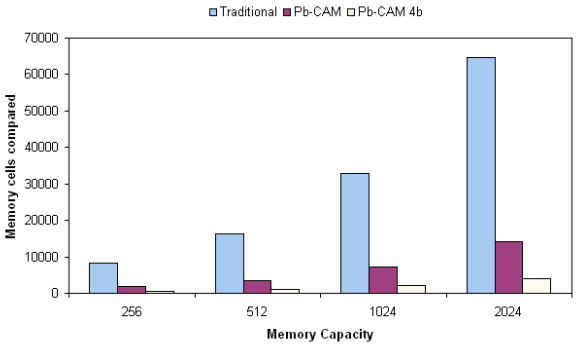

Fig. 11. Activated cells for a fixed word length (32b)

As can be seen in Figure 10, the traditional implementation shows a linear trend on the number of memory cells compared when the memory size is fixed (2048 positions) and the number of bits per word is ranged. On the contrary, the number of memory cells compared remains almost constant (logarithmic growth) for the PB-CAM implementation, due to the fact that the parameter size is $log(Nbits + 2)$. This graph also explains how for a reduced size of the word length, the parameter overhead could overcome the savings. The proposed implementation (PB-CAM 4b) improves these last results nearly by a factor equal to 4.

Figure 11 shows the linear tendency on the number of memory cells compared for the traditional CAM, the PB-CAM and the pipeline-banked PB-CAM when the word length is fixed (32b) and the memory size is ranged. The different slopes between the linear tendencies explain how for large sizes of memory, the savings with the PB-CAM are quite representative and the banked CAM still improves these results.

Finally, Figure 12 shows the distribution of the energy consumption in the pipelined PB-CAM. As can be seen, most of power consumption is still wasted in the pseudo-static logic (53%). Our research in this area has obtained further improvements in this energy consumption [10].

VII. Conclusions

Nowadays, the limiting factor in applications where the CAMs play a critical role is the power consumption of these devices. The integration levels achieved by current technology processes have turned the area and performance factors into secondary actors. Search-based applications with high-performance constrains, as IP routing processors, demand efficient implementations of content-addressable memories to cover the astringent constrains.

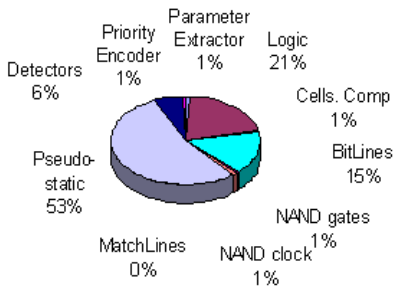

Fig. 12. Power distribution in the experimental pipeline.

The work presented in this paper has shown a practical implementation of a low-power CAM oriented to high-performance IP routing. The pipelined organization of the architecture promises greater scalability in the performance and density of IP routing processors.

Acknowledgements

This work was supported by the Spanish Ministry of Science and Education under contract TIC2003-07036.

References

[1] K. Pagiamtzis and A. Sheikholeslami, "A low-power content-addressable memory (CAM) using pipelined hierarchical search scheme," *IEEE Journal of Solid-State Circuits*, vol. 39, no. 9, pp. 1512–1519, September 2004.

[2] K.-J. Lin and C.-W. Wu, "A low-power CAM design for LZ data compression," *IEEE Transactions On Computers*, vol. 49, no. 10, pp. 1139–1145, October 2000.

[3] A. Efthymiou and J. D. Garside, "An adaptive serial-parallel CAM architecture for low-power cache blocks," in *IEEE International Symposium on Low Power Electronics and Design*, 2002, pp. 136–141.

[4] ——, "A CAM with mixed serial-parallel comparison for use in low energy caches," *IEEE Transactions On Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 3, pp. 325–329, March 2004.

[5] C.-S. Lin, J.-C. Chang, and B.-D. Liu, "A low-power precomputation-based fully parallel content-addressable memory," *IEEE Journal Of Solid-State Circuits*, vol. 38, no. 4, pp. 654–662, April 2003.

[6] C.-S. Lin, K.-H. Chen, and B.-D. Liu, "A low-power and low-voltage fully parallel content-addressable memory," in *IEEE International Symposium on Circuits and Systems*, 2003, pp. 373–376.

[7] I. Y.-L. Hsiao and D.-H. W. amd Chein-Wei Jen, "Power modeling and low-power design of content addressable memories," in *IEEE International Symposium on Circuits and Systems*, 2001, pp. 926–929.

[8] H. Miyatake, M. Tanaka, and Y. Mori, "A design for high-speed low-power CMOS fully parallel content-addressable memory macros," *IEEE Journal Of Solid-State Circuits*, vol. 36, no. 6, pp. 956–968, June 2001.

[9] P. Echeverría, J. L. Ayala, and M. López-Vallejo, "Application of a DVS Technique to a Banked Implementation of a PB-CAM," *Technical Report UPM-LSI-2005-003*, August 2005.

[10] ——, "A banked precomputation-based cam architecture for low-power storage-demanding applications," in *IEEE Mediterranean Electrotechnical Conference*, 2006.