Abstract

With current architectures, CAMs typically take significantly more area, power, and sometimes delay compared to location addressed memories of the same capacity. If these penalties are traded off against each other, there will be many new applications for CAMs that are not feasible or practical today. Our work is aiming to combine various CAM design methods used in literature which only aim to improve a single aspect of the problem, with our further improvements in a way to address multiple problems simultaneously to meet the requirements of today’s new applications. In this report, we overview the most effective methods found in our survey at a circuit level and an architectural level. By combining the current race scheme with precomputation and selective precharge we can achieve considerable power savings without sacrificing speed and area.

1 Introduction

Content addressable memories (CAM) have found applications in many areas from network routing to microprocessor cache tag lookups because they can offer a highly associative search. By allowing designers to address their data by its content rather than location has even enabled some more novel applications such as hardware compression/decompression [4] and alternate models of computation. Further adoption has been held back because compared to location addressed memories of the same capacity, CAMs typically take up significantly more area, power, and sometimes delay. If these penalties are traded off against each other, there may be some other applications for CAMs.

To harness the transistors provided by Moore’s Law, architects have turned to increasing core counts rather than increasing clock rates and instruction level parallelism because of power limitations. With the ability to fit many cores onto a single die comes the challenge of keeping all of their caches coherent. There are a variety of mechanisms to accomplish this, but a promising scalable technique uses reverse tagged directories. These can be implemented by CAMs, and by using a highly associative search the needed capacity can be reduced by orders of magnitude, making it reasonable to integrate on chip. Without the associative search the directory will require so much capacity that it will need to be implemented by off chip DRAM which will take more power and have higher delay.

In light of the application requirements for many-core cache coherence, this work will attempt to explore CAM designs appropriate for it. Much of the prior work on CAM design has focused on ternary CAMs which provide the ability for the stored data to contain search wildcards, but that is not needed for the intended application. Other past research has paid attention to reducing the active energy required for each search, but a directory may be inactive much of the time so it also is important its idle power is low. We will re-evaluate past approaches considering our different application desires and requirements. Our new contributions will be to consider some hybrid approaches for the entire system not previously attempted as well as exploring some new options for within the NAND based CAM cell.

2 Existing Solutions

In order to improve power consumption and performance of CAM cells, various design optimizations have been carried out at cell, system architecture and coding levels in CAM design. In this section, we will present the state-of-the-art design solutions that have been presented in the literature in a comprehensive frame that relates to our design goals.

2.1 Basics of Conventional CAM Cell Design

A CAM cell serves two basic functions: bit storage, as in RAM; and bit comparison, which is unique to CAM. Although there are various cell designs, for bit storage, typically a CAM cell uses an SRAM internally. The bit comparison function, which is logically equivalent to an XOR of the stored bit and the search bit, can be implemented two different approaches: the NOR cell and the NAND cell (Figure 1). Both approaches have their advantages, and there are various design optimizations that apply to each.

NOR CAM Cell

The NOR cell implements the comparison between the complementary stored bit, $D$ (and $\overline{D}$), and the comple-
NAND CAM Cell

The NAND CAM cell implements the comparison between the stored bit, D, and corresponding search data on complementary searchlines, (SL, SL'), using the three comparison transistors, \(M_1, M_D\) and \(M_T\), which are all typically minimum size to maintain high cell density. One can recognize node B as the PTL implementation of the XNOR function of inputs SL and D. In the case of a match, node B becomes high, turning transistor \(M_1\) on, which connects \(ML_n\) and \(ML_{n+1}\) in series. The mismatch case leaves node B low, disabling transistor \(M_1\) and disconnecting \(ML_n\) and \(ML_{n+1}\). The NAND nature of this cell is better understood when multiple cells are connected in series to for a CAM word, for which the ML resembles the pulldown path of a CMOS NAND gate.

The NAND search cycle starts with precharging ML with a PMOS transistor. Next, unlike the NOR cell, the NAND structure includes an NMOS evaluation transistor which is activated after precharge phase. In the case of a match, all XNORs in each CAM cell in the word evaluate to 1, and transistors \(M_1\) through \(M_n\) form a discharge path for ML. For the mismatch case, ML remains high. A sense amplifier detects the low (i.e. match) or high (i.e. miss) state charge to slightly above \(V_{DD}\) and trip their latch.
Figure 3: (a) Unpipelined NOR based matchline; (b) NOR based matchline segmented 4 ways [6] while mismatched ML lines will remain at a lower voltage, preserving the latch in initial state.

2.2.3 Selective Precharge & Pipelining

The previous ML techniques spend approximately the same amount of energy on every match line whether or not it is a match. When performing a search, if the first few bits do not match, the entire word can not match, so there is no point in checking the remaining bits [8]. Selective precharge initially searches only the first $n$ bits and only searches the remaining bits for words that matched in the first $n$ bits. With uniform random data, it will only have to search $\frac{1}{2^n}$ of the rows. For $n = 3$ this will save about 88% of the matchline power.

Once can also utilize pipelining [2,5], since generally an implementation may divide the matchline into any number of segments, where a match in a given segment results in a search operation in the next segment but a miss terminates the match operation for that word (Figure 3). If any stage misses, the subsequent stages are shut off, resulting in power saving. The drawbacks of this scheme are the increased latency and the area overhead due to the pipeline stages, which may be undesirable for some applications.

2.3 Search Line Driving Techniques

2.3.1 Hierarchical Searchlines

One method of saving searchline power is to shut off some searchlines (when possible) by using the hierarchical searchline scheme. The basic idea of hierarchical search lines is to exploit the fact that few matchlines survive the first segment of the pipelined matchlines. With the conventional searchline approach, even though only a small number of matchlines survive the first segment, all searchlines are still driven. Instead of this, the hierarchical searchline scheme divides the searchlines into a two-level hierarchy of global searchlines (GSLs) and local searchlines (LSLs). The GSLs are active every cycle, but the LSLs are active only when necessary (when at least one of the rows they cross is still active), saving power.

2.3.2 Eliminating Searchline Precharge

Eliminating the SL precharge phase reduces the toggling of the searchlines, and reduces power. As discussed above, matchline-sensing schemes that precharge the matchline low do not need SL precharge, since enabling the pull-down path in the NOR cell does not interfere with matchline precharge. Typically, about 50% of the search data bits toggle from cycle to cycle, hence there is a 50% reduction in searchline power, compared to the precharge-high matchline-sensing schemes that have an SL precharge phase.

2.4 Architectural Technique: Precomputation of Data

Up to now, we have covered circuit level ideas to improve CAM power consumption and performance. One can also introduce architectural level techniques to save power. Pre-computation stores some extra information along with each word that is used in the search operation to save power. The extra bits are derived from the stored word, and used in an initial search before searching the main word, and if the initial search fails, aborting the subsequent search, thus saving power.

One method as shown in Figure 4, uses a pre-computation circuit to count the number of ones, and stores this data along with the word [3]. When searching first the number of ones in each row is compared and those that do not match are not compared, but the most compelling part of PB-CAM results from exploiting the 1’s count to simplify the comparison circuitry in the second search stage. It can use simplified CAM cells that have only one pulldown path (a total of two pulldown transistors instead of four) which also eliminates one of the searchlines. Even with the simplified cell, the match result is always correct since a mismatched word in this second stage always has the same number of 1’s as the search data word, hence a mismatch will always cause at least one path to ground. The smaller cells should also save power because they put less capacitance on the matchline.

3 Basis of Comparison

To evaluate our various designs we will use HSpice with a modern process technology. We are most concerned with reducing the energy consumption, both active and idle,
however it is important we do this without unreasonably increasing the area or delay. Area will not be considered as strongly because we will be unable to optimally layout all of the designs considered, but fortunately our proposed designs should not drastically differ in area. We will need to make some approximations or layout some portions to get reasonable estimates for the parasitics of the long wires.

It will probably be unreasonable to build one of the proposed directories from a single CAM array, so it will have to be banked to make the desired capacity. To pick the optimum bank size, we will investigate how the various CAM designs do when scaled horizontally (wider rows) or vertically (more rows) to see how their energy and delay vary. More interestingly, some designs will probably do better relative to the other designs for certain size ranges.

Besides measuring the energy to perform a search, we will also compare the designs at different search rates to see how well they idle. To validate any of the energy saving techniques, we will also compare them against scaling the supply voltage. This work does not attempt to deal with the many complications of noise and process variation, but in a modern technology they are inevitable and must be accounted for. To keep our designs reasonable we will maintain decent noise margins and we will avoid structures that are dangerously susceptible to variation.

4 Proposed Design Work

As an initial baseline we will build many of the optimizations described in Section 2 to confirm previous results. These past optimizations have great potential, but many of them were tested in isolation. We believe a great design can emerge from a hybrid. More specifically, we would like to combine selective precharge (2.2.3) with precomputation (2.4) and a current racing matchline (2.2.2). First we will use a handful of NAND CAM cells at the front of each row to greatly reduce how many rows are activated for the later stages. Following this with the ones counting precomputation will turn off a couple more rows, but much more importantly it will allow us to use the smaller NOR CAM cells that need only one searchline per column. Finally rather than using the conventional precharge high matchline, we will use a current racing matchline. It may be possible to save even more energy using a low-swing matchline, but that may be much more susceptible to noise and variation.

One of the downsides to NAND CAM cells is that their delay is typically higher than their NOR counterparts. Since we intend to use NAND CAM cells to implement selective precharge, we will also consider some methods of improving their speed. Part of the delay from the NAND cells comes from their long series of NMOS transistors. To improve the speed of that NMOS ($M_D$), we would like to try overdriving the searchlines, which should allow more current to pass without having to make that NMOS wider. We could get a similar effect by making $M_D$ and $M_T$ lower $V_t$ if the process technology allows it. To combat some of the charge sharing problems of a NAND matchline we would like to try a current racing scheme instead of the conventional dynamic precharge high scheme. Some of these NAND CAM cell modifications may increase the cell’s area or power consumption, but it might not impact the overall system too much since most of our array will still be NOR CAM cells.

5 Results

The following table summarizes performance results obtained from literature of the CAM design methods covered in this report. The results in the table are simulated using HSpice for a 0.18 um CMOS process [6].

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Cycle Time (ns)</th>
<th>ML Energy (fJ/bit/search)</th>
<th>Model of Energy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conventional</td>
<td>3.9</td>
<td>9.5</td>
<td>$V_{DD} V_{MAX}$</td>
</tr>
<tr>
<td>Low Swing</td>
<td>3.1</td>
<td>4.2</td>
<td>$C_{ML} V_{DD} V_{MAX}$</td>
</tr>
<tr>
<td>Current Race</td>
<td>3.7</td>
<td>5.3</td>
<td>$C_{ML} V_{DD} V_{MAX}$</td>
</tr>
<tr>
<td>Selective Precharge &amp; Pipelining</td>
<td>5.8</td>
<td>5.0</td>
<td>$V_{DD} V_{ML} 2^k (V_t + V_{DD})$</td>
</tr>
</tbody>
</table>

Our design will use a clever combination of these methods with further improvements. Our current proposed architecture will use selective precharge at the front, followed by a current race scheme to reduce voltage swing on ML to $V_{tn}$, further assisted by precomputation method to reduce power on select lines. The expected model for our energy consumption on matchlines is $C_{ML} V_{DD} V_{tn} 2^k (V_t + V_{DD})$, where $n$ is number of bits in the word, and $k$ is the number of bits pre-searched in the word.

6 Conclusion

There has been a large amount of effort to improve CAM design in the past, and in this report we have summarized the most effective design techniques that we found in the literature, in a way that relates to our design goals in a comprehensive frame. Previous efforts have been trying to improve generally either just speed or area, resulting in advantages for a limited number of applications. With the background we summarized here, and our future work and design, we will try to explore the CAM design by utilizing a clever combination of the methods above, with added improvements to meet the design requirements of today’s applications, and even create new application areas for CAMs that were not feasible or practical with current design options.

References


