High-Performance Priority Encoder for Content Addressable Memories

Wilson W. Fung and Manoj Sachdev
University of Waterloo
Dept. of Electrical and Computer Engineering, 200 University Ave. West, Waterloo, Ontario, N2L 3G1
Tel: (519) 888-4567, E-mail: wwfung@vlsi.uwaterloo.ca

Abstract

This paper describes a sub-block level Priority Encoder for Content Addressable Memories. Simulations show that our design consumes only half of the energy as reported in [3] for the same worst-case delay and technology.

1. Introduction

Priority Encoder (PE) is a basic building block in many digital and mixed-signal systems. Unlike a simple encoder that allows only one of its inputs at logic-1 state, a PE does not have such restriction. It can resolve multiple logic-1 inputs and perform encoding according to the one with the highest priority. Applications of PE include resource arbitration, analog-to-digital conversion, and multiple match resolution in Content Addressable Memories (CAM). The former two require only small PEs with sizes up to 32-bit. However, the PE in large-capacity CAMs can have a resolution up to a hundred thousand inputs if the Match signal of every CAM word is considered.

Figure 1 shows the floorplan of a CAM with distributed PEs. A multi-level encoding tree is employed to collect all match signals, and to resolve the encoded address associated with the highest priority match. Since the PE determines the latency and the rate at which a CAM can output data, its design is critical to the overall CAM performance. Recent publications [1]-[2] on CAM have mainly focused on power reduction in cell array and match-line sensing, while little attention has been made toward the priority encoding circuits for CAM. In this paper, the design of a high-performance sub-block level PE for CAM is presented.

2. Previous Work on Priority Encoding

The structure of a PE is similar to that of any datapath circuits. In general, there are two ways to achieve priority encoding. They are illustrated in Figure 2. Previous work of PE [3]-[4] are all based on the "resolve-then-encode" approach. In this scheme, priority encoding is divided into two steps: Multiple Match Resolution (MMR) and Address Encoding (AE). The MMR in [3] is implemented based on simple logic equations using NP domino for both critical and non-critical paths. Such design generates huge load on clock drivers. Cascading it with other blocks in CAM is also a challenge, due to its intrinsic glitches at the outputs. The PE in [4] exhibits the same shortcomings, with the addition that it is not even scalable beyond 32-bit.

3. A "Direct Encode" Priority Encoder

The design of our sub-block level PE is based on the direct-encode approach (Figure 2). This scheme simplifies the control signals due to the integration of the "resolve" and "encode" stages. Another advantage is its potential to be pipelined efficiently, as a result of the hierarchical and balanced partitioning in our design. Deeper pipelining is one of the trends to increase CAM throughputs. Apparently the direct-encode PE exhibits such advantage.

A diagram of our sub-block level PE is illustrated in Figure 3. This 256-bit PE employs a four-level encoding tree. Each level contains four lower-level blocks. The encoded address bits, from the lower-level blocks, are driven by tri-state inverters, and then multiplexed into the same local bus. In the first level, the address bits can be described as:

$$A1 = ML0' \bullet ML1'$$
$$A0 = ML0' \bullet (ML1 + ML2')$$

Note that the above logic equations are simplified versions of the originals. They are based on the assumption that at least one input of every block is logic-1 during evaluation. Such assumption is valid because the erroneous encoded bits will be ignored by the upper-level stage. This observation greatly simplifies the logics, and leads to lower capacitance and better performance. The same logic described above can be applied to the upper-level stages. In addition, look-ahead (LA) signals are embedded in each stage to speed up the arbitrations among the tri-state drivers.

In our design, only the LA paths employ domino logics. The rest of the encoding tree is implemented using static CMOS logic gates. This reduces the capacitive load on the clock drivers substantially as compared to [3]. To avoid redundant switching at the on-set of pre-charge, Set-dominant Latches (SDL) are used as the dynamic-to-static interface. SDL (Figure 4) catches the monotonic 1-to-0 transition and offers the lowest overhead on domino paths.

4. Results and Conclusions

Our PE has been simulated with extracted parasitics using HSPICE. The energy-delay plot is shown in Figure 5. For the same latency constraint, our design consumes only half of the energy as report in [3]. The load of our PE on the clock drivers is only one-fifth as compared to [3]. These results show that our "direct-encode" PE is suitable for low-power and high-performance CAMs.

5. References

[1] I. Arsovski, T. Chandler, and A. Sheikholeslami, "A Ternary Content-Addressable Memory (TCAM) based on 4T static storage and including a current-race sensing scheme," *IEEE J. Solid-State Circuits*, vol. 38, no. 1, pp. 155-158, January 2003

[2] C. Lin, J. Chang, and B. Liu, "A Low-power Precomputation-based Fully Parallel Content-Addressable Memory," *IEEE J. Solid-State Circuits*, vol. 38, no. 4, pp. 654-662, April 2003

[3] C. Huang, J. Wang, and Y. Wang, "Design of High-Performance CMOS Priority Encoders and Incrementer/Decrementers Using Multilevel Lookahead and Multilevel Folding Techniques," *IEEE J. Solid-State Circuits*, vol. 37, no. 1, pp. 63-76, January 2002

[4] J. Delgado-Frias and J. Nyathi, "A High-Performance Encoder with Priority Lookahead," *IEEE Trans. on Circuits and Systems - I*, vol. 47, no. 9, pp.1390-1393, Sept 2000

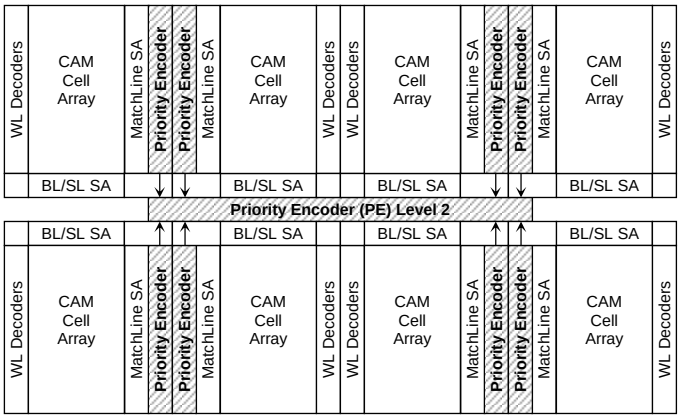

Figure 1: Floorplan of a sample CAM chip

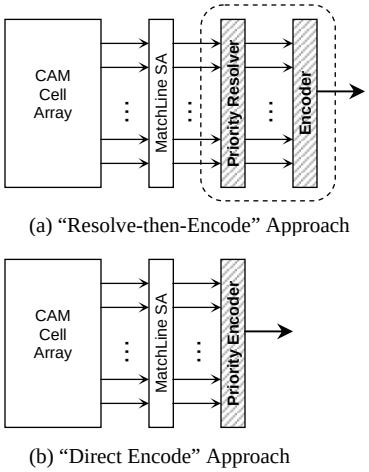

Figure 2: Two types of Priority Encoders

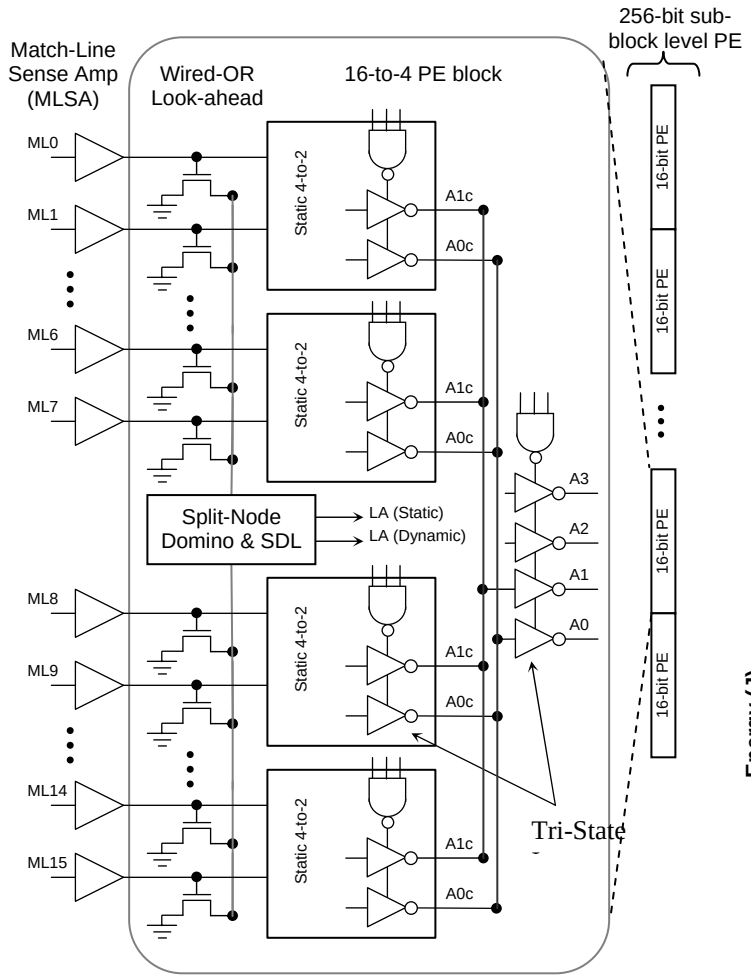

Figure 3: Block diagram of a 16-bit Direct-Encode PE

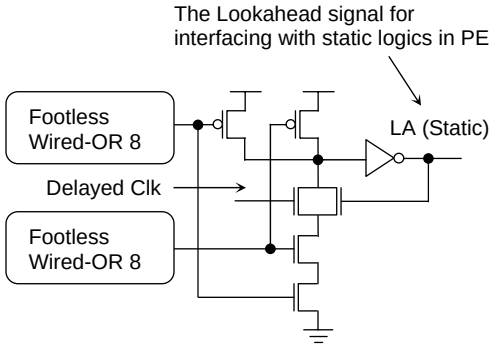

Figure 4: A Split-Node Domino-to-Static Interface

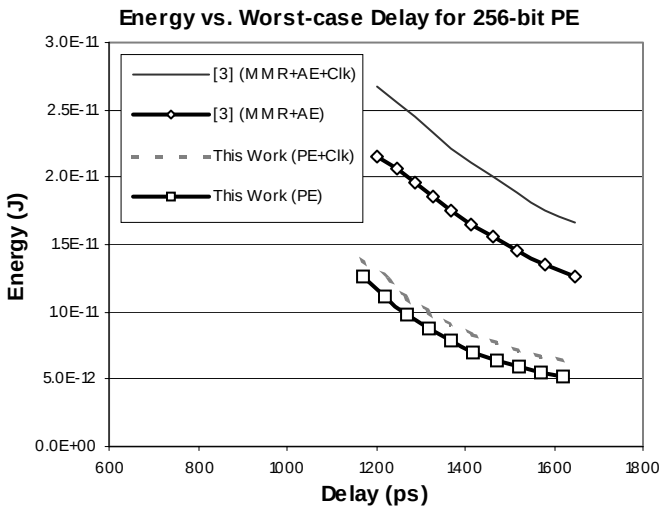

Figure 5: Energy vs. Worst-case Delay plot