International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 06 Issue: 03 | Mar 2019 www.irjet.net p-ISSN: 2395-0072

A SURVEY ON ENCODE-COMPARE AND DECODE-COMPARE ARCHITECTURE FOR TAG MATCHING IN CACHE MEMORY USING ERROR CORRECTING CODES

K.L. Deepa priya[1], V. Chandrasekaran[2], M. Parimala Devi[3], S. Pavithra[4]

1. Second Year- M.E. (Applied Electronics), Velalar College of Engineering and Technology, Erode, Tamil Nadu, India.
2. Professor and HOD, Medical Electronics department, Velalar College of Engineering and Technology, Erode, Tamil Nadu, India.
3,4. Assistant Professor (Sl.Gr.), Department of ECE, Velalar College of Engineering and Technology, Erode, Tamil Nadu, India.

Abstract- *Modern microprocessor performance will be improved, if the cache memories serve as accelerators. Due to technology scaling, caches are unarmed to soft errors. In tag matching cache, there are different types of architectures used and they are encode-compare architecture and decode-compare architecture based on direct compare method. This paper compares the encode-compare and decode-compare architecture using various performance metrics.*

Keywords:
Tag matching, Cache memory, Hamming distance, Error correcting codes, Translation Look aside Buffer (TLB).

1. INTRODUCTION

In cache tag matching, currently microprocessor caches are set-associative caches. A set associative cache has a tag directory and a data array. The tag directory stores tag addresses which are used to indicate which part of memory is stored in the data array. When an access is made to the cache, the set-address portion of the entire address is used to index into the tag directory and a set of tags (the number of tags read depends on the associativity) are read.

These tags are compared with the tag field of the incoming address to see if there is a match. When the entry is valid and a retrieved tag matches the incoming address tag field, then there is a "cache hit". If the incoming address tag field does not match with any of the stored tag, a "cache miss" happens.

Hamming code is one of the popular techniques based on forward error correction [1]. The recent computer employs Error-Correcting codes (ECC) to protect data and improve reliability [2]-[5]. Error detection is that the detection of errors caused by noise or different impairments throughout transmission from the transmitter to the receiver.

Error correction is that the detection of errors and reconstruction of the initial, error-free data. Good error management performance needs the theme to support the characteristics of the communication.

Common channel models embrace memory-less models wherever errors occur willy-nilly and with a particular chance and dynamic models wherever errors occur primarily in bursts.

Consequently, error-detecting and correcting codes are often typically distinguished between random-error-detecting/correcting and burst-error-detecting/correcting. Some codes may also be appropriate for a mix of random errors and burst errors.

When the data rate cannot be determined or it is highly variable, an error-detection scheme may be combined with a system for retransmission of erroneous data. This is referred to as Automatic Repeat Request (ARQ) and is most notably utilized in the web.

An alternate approach to error control is a Hybrid Automatic Repeat Request (HARQ), which is a combination of ARQ and error-correction coding.

Error-Correcting Codes area unit sometimes distinguished between convolutional codes and block codes:

- **Convolutional codes** are processed on bit-by-bit basis. They are significantly appropriate for implementation in

hardware and therefore the Viterbi decoder permits best secret writing.

- **Block codes** are processed on block-by-block basis. Repetition codes, hamming codes and multi-dimensional parity-check codes are the early examples of Block codes. They were followed by variety of economical codes, Reed–Solomon codes being the most notable due to their current widespread use. Turbo codes and Low-Density Parity-check Codes (LDPC) area unit comparatively new constructions that may offer nearly best potency.

2. BASIC DEFINITIONS

2.1 Hamming distance

The Hamming distance between 2 strings of equal length is that the range of positions at that the corresponding symbols square measure totally different.

In a different way, it measures the minimum range of substitutions needed to vary one string into the opposite or the minimum range of errors that would have reworked one string into the other.

Examples

The Hamming distance between:

- "Karolin" and "Kathrin" is 3.
- "Karolin" and "Kerstin" is 3.
- 1011101 and 1001001 is 2.
- 2173896 and 2233796 is 3.

2.2 Saturation adder

Saturation adder could be a version of arithmetic during which all operations like addition and multiplication are restricted to a set vary between a minimum and most worth.

For example, if the valid vary of values is from -100 to one hundred, the subsequent operations manufacture the subsequent values:

- 60 + 30 = 90
- 60 + 43 = 100

Figure 1 shows the (4, 2) saturation adder diagram.

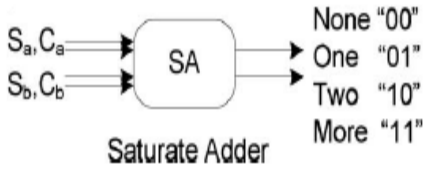

Fig-1: (4,2) Saturation Adder

By logic reduction, the logic equation for this (4, 2) saturating adder is given by,

$$Carry=C_a+C_b+S_aS_b$$

$$Sum=S_a+S_b+C_aS_a+C_bS_b$$

2.3 Encoder

A simple encoder circuit can receive a single active input out of 2^n input lines and generate binary code on *n* parallel output lines. For example, single bit 4 to 2 encoder takes in 4 bits and outputs 2 bits and it is shown in Figure 2.

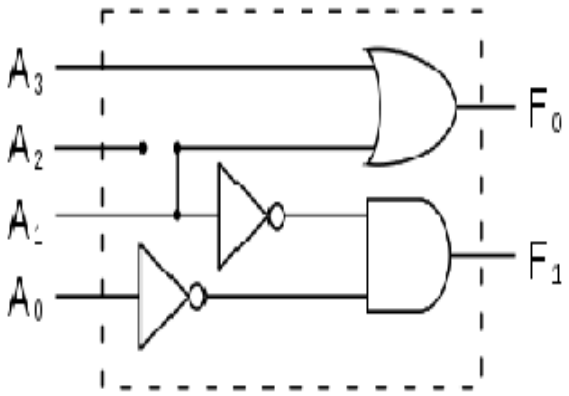

Fig-2: 4 to 2 Encoder

2.4 Decoder

It is a combinational circuit that converts the binary information from *n* input lines to a maximum of 2^n unique output lines and it is shown in Figure 3.

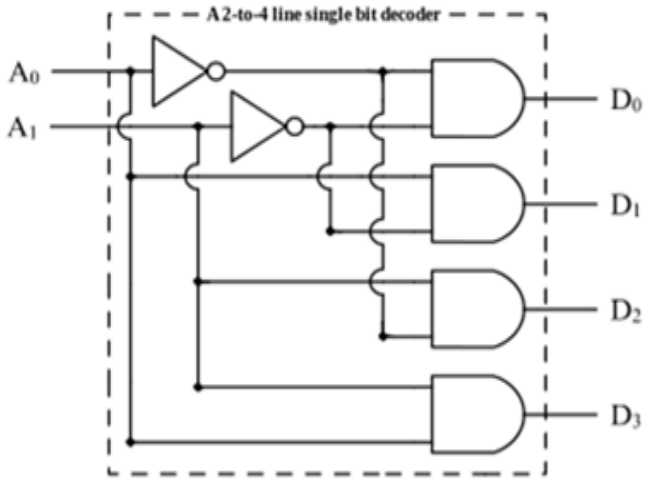

Fig-3: 2 to 4 Decoder

3. ENCODE AND COMPARE ARCHITECTURE

Decoding takes longer time than secret writing because it encompasses a series of error detection or syndrome calculation and error correction. To resolve the drawbacks of the decode and compare architecture, the decoding of a retrieved codeword is replaced with the encoding of an incoming tag in the encode-and-compare architecture as shown in Figure 4.

More precisely, a k-bit incoming tag is first encoded in the corresponding n-bit codeword and compared with the retrieved codeword as shown in Figure 4.

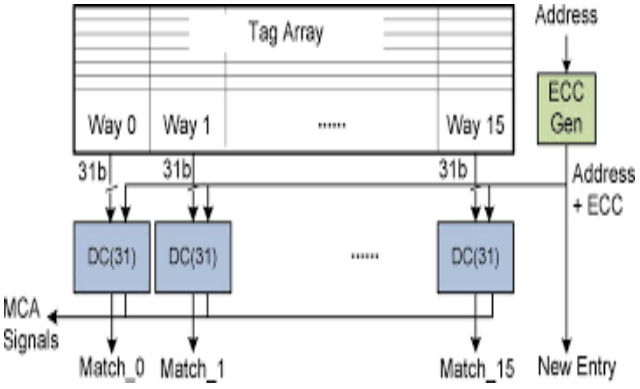

Fig -4: Encode and Compare Architecture

The comparison is to look at what number bits the 2 codewords dissent, not to check if the two codewords are exactly equal to each other. For this, there is a tendency to work out the overacting distance d between the 2 codewords and classify the cases consistent with the distance of 'd'.

Let t_{max} and r_{max} denote the numbers of maximally correctable and detectable errors, respectively. The cases are summarized as follows.

- If d = 0, X matches Y exactly.
- If 0 <d ≤ t_{max}, X will match Y provided at most t_{max} errors in Y are corrected.
- If t_{max}<d ≤ r_{max}, Y has detectable but uncorrectable errors. In this case, the cache may issue a system fault, so as to make the central processing unit take a proper action.
- If r_{max}<d, X does not match Y

Assuming that the incoming address has no errors, 2 tags are regarded as matched if 'd' is in either the primary or the second ranges. In this way, while maintaining the error-correcting capability, the architecture can remove the decoder from its critical path at the cost of an encoder being newly introduced.

Note that the encoder is, in general, much simpler than the decoder, and thus the encoding cost is significantly less than the decoding cost.

Since the above method needs to compute the Hamming distance, presented a circuit dedicated to the computation.

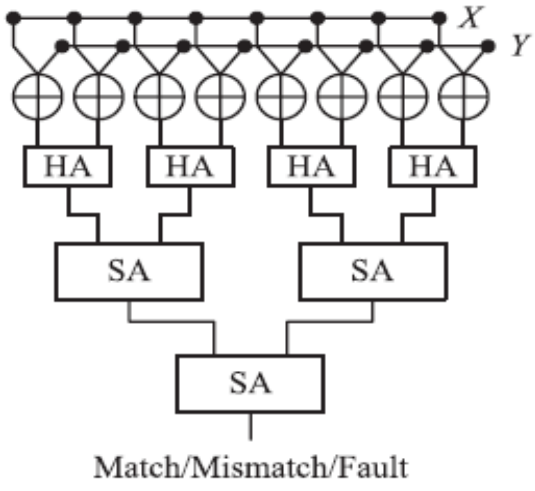

Fig -5: SA based architecture supporting the direct compare method

The circuit shown in Figure 5 performs XOR operations for each try of bits in X and Y thus on generate a vector representing the bitwise distinction of two codewords. The following half adder (HA) is used to count the number of 1's in two adjacent bits in the vector.

The numbers of 1's is accumulated by passing through the following SA tree [6]. In the SA tree, the accumulated value z is saturated to r_{max}+ 1 if it exceeds rmax. More precisely, given inputs x and y, z can be expressed as follows:

$$Z = \begin{cases} A + B, & \text{if } A + B \le r_{max} \\ r_{max} + 1, & \text{else} \end{cases} (1)$$

The final accumulated value indicates the range of *d*. As the compulsory saturation necessitates additional logic circuitry, the complexity of a SA is higher than the conventional adder.

4. DECODE AND COMPARE ARCHITECTURE

A cache memory is considered where a k-bit tag is stored in the form of an-bit codeword after being encoded with a (n, k) code. In the decode-and-compare architecture, the n-bit retrieved codeword should first be decoded to extract the original k-bit tag. The extracted *k*-bit tag is then compared to the *k*-bit tag field of an incoming address to determine whether the tags are matched or not.

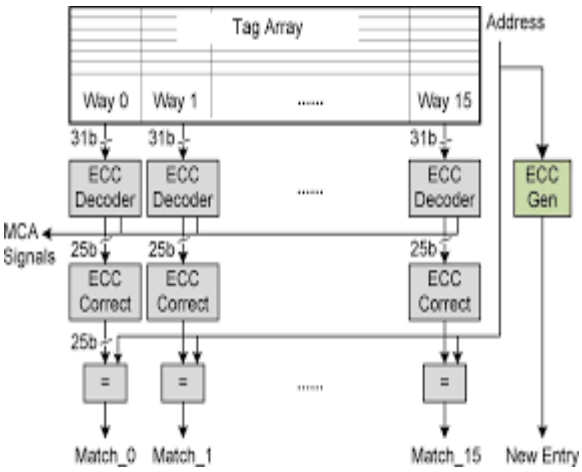

Fig -6: Decode and Compare Architecture

As the retrieved codeword should go through the decoder before being compared with the incoming tag, the critical path is too long to be employed in a practical cache system designed for high-speed access. Since the decoder is one of the most complicated processing elements, in addition, the complexity overhead is not negligible. The decode compare architecture [8] is shown in Figure 6.

5. DISCUSSION

Results are discussed in the table shown below.

ENCODE-COMPARE ARCHITECTURE	DECODE-COMPARE ARCHITECTURE
Encoder is simpler than decoder	Decoder is complex than encoder
TheComplexity overhead is negligible.	The Complexity overhead is not negligible.
In encode-compare architecture, the decoder is removed by introducing an encoder.	For high speed access, the decoder will increase the critical path.
Encoding cost is significantly less.	Decoding cost is more.

6. CONCLUSION

Decoder is complex than encoder and the complexity overhead is not negligible. Decoding cost is more than encoding cost. Decoder will increase the critical path for high speed access.

Encoder is simpler than decoder and the complexity overhead is negligible. In Encode-compare architecture, the decoder is removed by introducing an encoder. Encoding cost is significantly less than decoding cost.

Comparing encoder and decoder architecture, encoder architecture is the most efficient one and it has less complexity and less cost.

REFERENCES

[1] Aswathi D and Keerthi Sahadevan, "An Efficient Low Complexity Low Latency Architecture for Matching of Data Encoded with Error Correcting Code Using a Cache Memory", Int. Journal of Engineering Research and Application, Vol. 6, Issue 10, (Part -3) October 2016, pp.82-85.

[2] Chang, J., Huang, M., Shoemaker, J., Benoit, J., Chen, S.L., Chen,W., Chiu, S., Ganesan, R., Leong, G., Lukka, V., Rusu, S. and Srivastava, D. (2007),''The 65-nm 16-MB shared on-die L3 cache for the dual-core Intel xeon Processor 7100 series'', IEEE J. Solid-state Circuits, Vol. 42, No. 4, pp. 846–852.

[3] Warnock, J.D., Chan, Y. H., Carey, S.M., Wen, H., Meaney, P.J., Gerwig, G., Smith, H.H., Chan, Y.H., Davis, J., Bunce, P., Pelella, A., Rodko, D., Patel, P., Strach, T., Malone, D., Malgioglio, F., Neves, J., Rude, D.L. and Huott, W.V. (2012), ''Circuit and physical design implementation of the microprocessor chip for the Enterprisesystem'',IEEE J. Solid-state Circuits, Vol. 47, No. 1, pp. 151–163.

[4] Ando, H., Yoshida, Y., Inoue, A., Sugiyama, I., Asakawa, T., Morita, K., Muta, T. And Motokurumada, T., Okada, S., Yamashita, H., and Satsukawa, Y. (2003), '' A 1.3 GHz fifth generation SPARC64 microprocessor'',IEEEISSCC. Dig. Tech. Papers, pp. 246–247.

[5] Tremblay and M. Andchaudhry, S. (2008), ''A third-generation 65nm 16-core 32-thread plus 32-Scout-thread CMT SPARC processor. ISSCC. Dig. Tech. Papers, pp. 82–83.

[6] Wu, W., Somasekhar, D. and Lu, S.L. (2012), ''Direct compare of information coded with error-correcting codes'', IEEE Trans. Very Large Scale Integr. (VLSI) Syst., Vol. 20, No. 11, pp. 2147–2151.

[7] Pedro Reviriego, Salvatore Pontarelli, Juan Antonio Maestro, and Marco Ottavi (2013), ''A Method to Construct Low Delay Single Error Correction Codes for Protecting Data Bits Only'', IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 32, No. 3.

[8] Mary Francy Joseph and Anith Mohan (2016), ''Improved Architecture for Tag Matching in Cache memory Coded with Error Correcting Codes'', Global Colloquium in Recent Advancement and Effectual Researches in Engineering, Science and Technology (RAEREST 2016), pp.544 – 551.