Low-Power Architecture for Reliability of Data Encoded With Error Correcting Codes

Satyakiran Kambhampati
GMR Institute of Technology

***Abstract*—** A new design architecture for coordinating the information ensured with an error-correcting code (ECC) is introduced in this paper to reduce complexity and power consumption. In view of the way that the code word of an ECC is normally represented as combination of input data and parity bits. The proposed design parallelizes the comparison of the raw information and that of the parity data. Further another butterfly-shaped weight accumulator (BWA) is introduced for the exact calculation of the Hamming distance. Based on the BWA, the proposed design analyzes whether the incoming information is same as to that of the cached information or not. For a 16-bit message signal the presented design architecture lessens the power and delay. For this purpose bubbled NAND logic (equivalent to Or-gate tree) was used in the implementation part for the reduction of delay. All the simulation results incorporated in this paper was done by using Xilinx software 14.5.

Keywords*—Data comparison, error-correcting codes (ECCs), Hamming distance, bubbled- logic.*

I. INTRODUCTION

Information comparison is broadly utilized as a part of processing frameworks to perform numerous operations, for example, the tag matching in a cache memory and the virtual-to-physical address translation in a translation look aside buffer (TLB). On account of such commonness, it is essential to execute the comparison circuit with low equipment complexity. Moreover, the information comparison as a rule dwells in the critical path of the components that are formulated to improve the overall system performance, e.g., caches and TLBs, whose outputs decide the stream of succeeding operations in a pipeline. The circuit, subsequently, must be intended to have a low latency as could be expected under the circumstances otherwise the parts will be excluded from filling in as accelerating agents and the general execution of the entire framework would be seriously disintegrated. As recent computers utilize error correcting codes (ECCs) to secure information and enhance reliability [1]–[5], complicated decoding methodology, which must go before the data comparison, lengthens the critical path and compounds the complexity overhead. In this way, it turns out to be considerably harder to meet the above design constraints. Notwithstanding the requirement for modern plans as depicted, the works that adapt to the issue are not generally known in the set off the literature since it has been usually treated within industries for their products. But recently it was gained of an ever increasing number of considerations from the academic field.

The most recent solution for the matching problem is the direct compare method [6], which encodes the incoming information and then compares it with the retrieved data that has been encoded as well. Hence this method eliminates the complex decoding mechanism from the critical path. In performing the comparison, the method does not examine whether the retrieved data is exactly same as the incoming data.

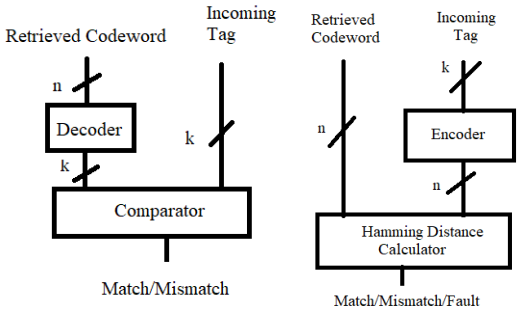

Fig.1. (a) Decode-and-compare model and (b) encode-and-compare model

Rather, it checks if the recovered information lives in the error correctable scope of the codeword relating to the incoming information. As the checking requires an extra circuit to process the Hamming distance, i.e., the quantity of various bits between the two code words, the saturate adder(SA) was presented [6] as an essential building block for computing the Hamming distance. In any case, [6] did not consider a vital certainty that may enhance the viability further, a general ECC codeword is normally represented in a systematic form in which the data and parity parts are totally isolated from each other [7]. Also, as the SA always forces its output not to be greater than the number of detectable errors by more than one, it adds to the increase of the whole circuit complexity.

In this connection, we remodel the SA-based direct compare architecture to lessen the idleness and equipment complexity nature by settling the previously mentioned downsides. More particularly, we consider the characteristics of systematic codes outlining the proposed architecture and propose a low-complexity nature processing component that computes the Hamming distance quicker. Thusly, the latency and the hardware equipment complexity natures are diminished impressively even compared with the SA-based design. The rest of this paper is organized as follows.
Section II reviews previous works. The proposed architecture is explained in Section III, and evaluated in Section IV. Finally, concluding remarks are made in Section V.

II. LITERATURE SURVEY

The theory of Error Correcting Codes was originated in the late 1940's by Richard Hamming, mathematician who worked for Bell Telephone. Hamming's motivation was to program computer to correct errors which arose in punch-card programs. Hamming's overall motivation behind the theory of Error Correcting Codes was to reliably enable digital communication. Data comparison circuit is a logic that has many applications in a computing system. For example, to check whether a piece of information is in a cache, the address of the information in the memory is compared to all cache tags in the same set that might contain that address. Another place that uses a data comparison circuit is in the Translation Look-aside Buffer (TLB) unit. TLB is used to speed up virtual to physical address translation. Error correcting codes (ECC) are widely used in modern microprocessors to enhance the reliability and data integrity of their memory structures. For example, caches on modern microprocessors are protected by ECC. If a memory structure has been protected with ECC, a piece of data is encoded first and the entire codeword including the ECC check bits are written into the memory array.

A. Decode-and-Compare Architecture

Let us consider a cache memory where a k-bit tag is stored always as a n-bit codeword in the form of being encoded by a (n, k) code. In the decode and compare model shown in Fig. 1(a), the n-bit recovered codeword should first be decoded to extract the first k-bit tag. The extracted k-bit tag is then compared with the k-bit tag field of an incoming address to decide whether the tags are matched or not. As the recovered codeword should go through the decoder before being compared with the incoming tag, the critical path is too long to to be employed in practical cache system intended for fast access. Since the decoder is a standout amongst the most complicated processing components, likewise, the complexity overhead isn't negligible.

B. Encode-and-Compare Architecture

Note that decoding is generally more complex and takes additional time than encoding as it incorporates a series of error detection or syndrome calculation and error correction [7]. The implementation brings about [8] support the claim. To determine the disadvantages of the decode-and-compare architecture, subsequently, the decoding of a recovered codeword is supplanted with the encoding of an incoming tag in the encode-and-compare architecture All the more absolutely, a k-bit incoming tag is first encoded to the relating n-bit codeword X and compared with an n-bit recovered codeword Y as appeared in Fig. 1(b). The comparison is to inspect what number of bits the two codewords vary, not to check if the two codewords are precisely equivalent to each other. For this, we find the Hamming distance d between the two codewords and characterize the cases as per the scope of d. Let t_{max} and r_{max} signify the quantities of maximally correctable and noticeable mistakes, individually. The cases are summarized as takes after.

1) If $d = 0$, X matches Y exactly.
2) If $0 < d \leq t_{max}$, X will match Y provided at most t_{max} errors in Y are corrected.
3) If $t_{max} < d \leq r_{max}$, 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.
4) If $r_{max} < d$, X does not match Y.

Assuming that the incoming location has no errors, we can view the two tags as coordinated if d is in either the first or the second ranges. Along these lines, while keeping up the error correcting capability, the design can expel the decoder from its critical path at the cost of an encoder being recently presented. Note that the encoder is, in general ,much simpler than decoder and consequently the encoding cost is fundamentally not as much as the decoding cost.

III. DESIGN ARCHITECTURE

This section presents a new architecture that can reduce the latency and complexity of the data comparison by using characteristics of systematic codes. Finally, a new combinational logic element was added to improve the performance.

A. Data Path Design for Systematic Codes

In the SA-based architecture [6], the comparison of two codewords is conjured after the incoming tag is encoded. In this way, the critical path comprises of a series of the encoding and the n-bit comparison as appeared in Fig. 2(a). Notwithstanding, [6] did not consider the way that, practically speaking, the ECC codeword is of a systematic form in which the information and parity parts are totally separated. As the data part of a systematic codeword is precisely same as the incoming tag field, it is instantly accessible for comparison while the parity part becomes available simply after the encoding is finished. Grounded on this reality, the comparison of the k-bit tags can be begun before the remaining (n– k) - bit comparison of the parity bits. In the proposed architecture, in this manner, the encoding procedure to produce the parity bits from the incoming tag is performed in parallel with the tag comparison, decreasing the general latency as appeared in Fig. 2(b).

B. Architecture for Computing the Hamming Distance

The architecture grounded on the data path design is shown in Fig. 3[9]. It contains multiple butterfly-formed weight accumulators (BWAs) proposed to improve the latency and complexity of the Hamming distance computation. The basic function of the BWA is to count the number of 1's among its input bits. It consists of multiple stages of HAs as shown in Fig. 4(a), where each output bit of a HA is associated with a weight.

The HAs in a stage are connected in a butterfly shape to accumulate the carry bits and the sum bits of the upper stage independently. In other words, the two inputs of a HA in a stage, with the exception of the first stage, are either carry bits

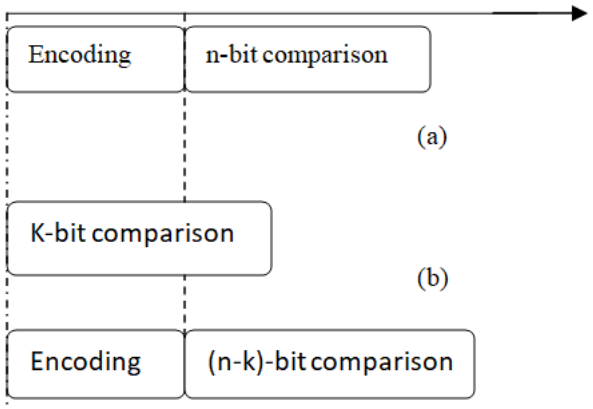

Fig.2. Timing diagram of the tag match in (a) direct compare method (b) Proposed architecture

or sum bits processed in the upper stage. This connection technique prompts a property that if an output bit of a HA is set, the number of 1's among the bits in the paths reaching the HA is equivalent to the weight of the output bit.

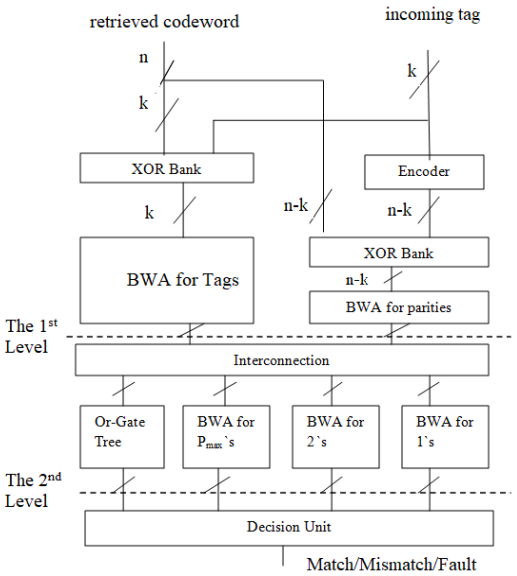

Fig.3.Architecture optimized for systematic codewords

Since what we require isn't the exact Hamming distance but the range it belongs to, it is possible to simplify the circuit. At the point when r_{max} = 1, for instance, at least two or more than two 1's among the input bits can be viewed as a similar case that falls in the fourth range. In such a case, we can replace a few HAs with a straightforward bubbled NAND-gate tree as shown in Fig. 4. This is an advantage over the SA based architecture. Instead of using Or-gate tree as in [9] previous literature using bubbled NAND-gate tree gives further improvements in terms of delay.

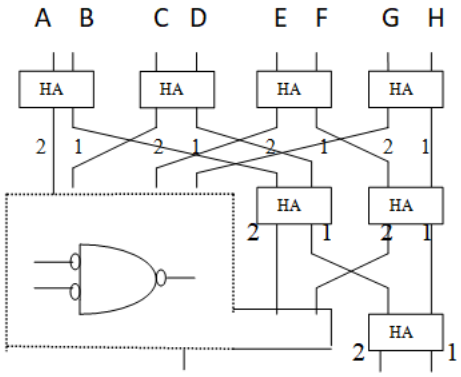

Fig.4. Proposed BWA Revised structure for matching of ECC-protected data

Note that in Fig. 4, there is no overlap between any pair of two carry bit lines or any pair of sum bit lines. We now clarify the general design in more detail. Each XOR organize in Fig. 3 produces the bitwise difference vector for either information bits or parity bits, and the accompanying processing elements count the quantity of 1's the vector, i.e., the Hamming distance. Each BWA at the first level as shown in Fig.4, and produces an output from the OR-gate tree and a several weight bits from the HA trees. In the interconnection, such outputs are fed into their related processing components at the second level. The output of the OR-gate tree is connected with the subsequent OR-gate tree at the second level, and the rest of the weight bits are associated with the second level BWAs as indicated by their weights. Clearly, the bits of weight w are connected to the BWA responsible for w-weight inputs. Each BWA at the second level is related with a weight of a power of two that is less than or equal to P_{max}, where P_{max} is the largest power of two that is not more than r_{max}+ 1. As the weight bits related with the fourth range are all ORed in the modified BWAs, there is no compelling reason to manage the powers of two that are bigger than P_{max}.

Let us consider a simple (8, 4) single-error correction double-error detection code. The corresponding first and second level circuits are shown in Fig.5. Note that the encoder and XOR banks are not drawn in Fig.5.for the sake of simplicity. Since r_{max}= 2, P_{max}= 2 and there are only two BWAs dealing with weights 2 and 1 at the second level. As the bits of weight 4 fall in the fourth range, they are ORed.
The remaining bits associated with weight 2 or 1 are connected to their corresponding BWAs. Note that the interconnection induces no hardware complexity, since it can be achieved by a bunch of hard wiring.

Taking the outputs of the behind circuits, the decision unit decide whether the incoming tag matches with the retrieved codeword by considering the four ranges of the Hamming distance. The decision unit is actually a combinational logic of which functionality is indicated by a truth table that takes the outputs of the first circuits as sources of info. For the (8, 4) code that the relating first and second level circuits are given in Fig. 5, the functionality table for the choice unit is illustrated in Table I. Since U and V can't be set at the same

time, such cases are certainly incorporated as don't care terms in table I.

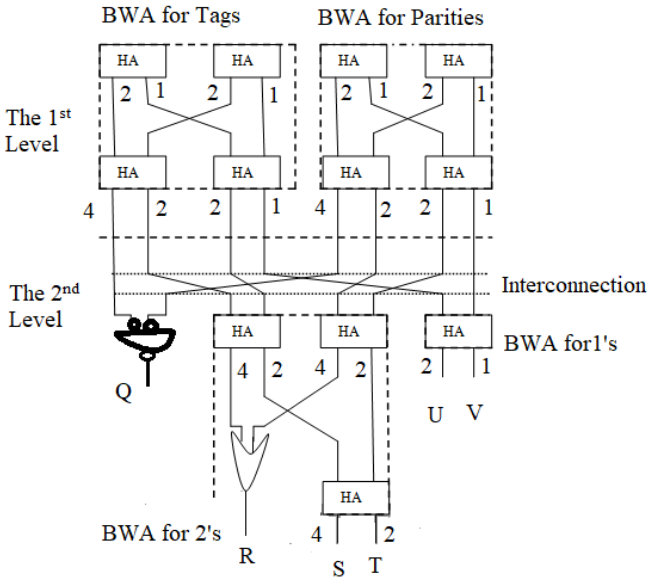

Fig.5.First and second level circuits for (8, 4) code

TABLE I. Truth Table of the Decision Unit

Q OR R OR S	T	U	V	Decision Unit
0	0	0	x	Match
0	0	1	x	Fault
0	1	0	0	Fault
0	1	0	1	Mismatch
0	1	1	x	Mismatch
1	x	x	x	Mismatch

C. General Expressions for the Complexity and Latency

The complexity and latency of combinational circuits were highly influenced by the algorithm which is employed. Furthermore, the complexity and latency are normally contradictory with each other, so it is unfortunately difficult to infer a logical and completely deterministic condition which gives the relation between the number of gates and the latency for the proposed design and furthermore for the ordinary SA-based architecture. To go around the trouble in diagnostic inference, we show some variables for the non deterministic parts rather an expression to estimate the latency and complexity. The complexity of the proposed design C can be represented as

$$C = C_{XOR} + C_{ENC} + C_{BWA}(k) + C_{BWA}(n-k) + C_{2nd} + C_{DU}$$
$$\leq n + C_{ENC} + 2\,C_{BWA}(n) + C_{DU} \quad (1)$$

Where C_{XOR}, C_{ENC}, C_{2nd}, C_{DU}, and $C_{BWA}(n)$ are the complexities of XOR banks, an encoder, the second level circuits, the decision unit and a BWA for n inputs respectively. Using recursive relation $C_{BWA}(n)$[9] can be calculated as

$$C_{BWA}(n) = C_{BWA}(\lfloor n/2 \rfloor) + C_{BWA}(\lceil n/2 \rceil) + 2\lfloor n/2 \rfloor \quad (2)$$

Where the seed value $C_{BWA}(1)$is zero. Note that when a+b =c. $C_{BWA}(a) + C_{BWA}(b) \leq C_{BWA}(c)$ holds for all positive integers a, b and c. Because of inequality and fact that an OR-gate tree for n inputs is always simpler than a BWA for n inputs, both $C_{B\,WA}(k) + C_{B\,WA}(n-k)$ andC_{2nd} are bounded by $C_{BWA}(n)$. The latency of the proposed architecture[9], L, can be expressed as

$$L \leq \max [L_{XOR} + L_{BWA}(K), L_{ENC} + L_{XOR} + L_{B\,WA}(n-k)] + L_{2nd} + L_{DU}$$
$$\leq \max (1+\lceil \log_2 k \rceil, L_{ENC} + 1 + \lceil \log_2 (n-k) \rceil) + \lceil \log_2 n \rceil + L_{DU} \quad (3)$$

Where L_{XOR}, L_{ENC}, L_{2nd}, L_{DU} and $L_{BWA(n)}$ are the latencies of a XOR bank, an encoder, the second level circuits, the decision unit and a BWA for n inputs respectively. Note that the latencies of the OR-gate tree and BWAs for $x \leq n$ inputs at the second level are bounded by $\lceil \log_2 n \rceil$.

IV. EXPERIMENTAL RESULTS

The simulation results for encoding and decoding of the codeword for the proposed architecture both the transmission as well as receiver sections were shown in the fig.6 and fig.7. These simulation results were obtained by using ISE simulator of XILINX software version 14.5.

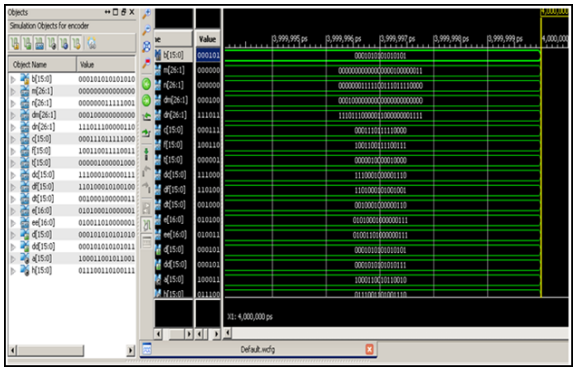

Fig.6. Simulated output

The denoted notations the above figures are described below. In the design module ‘a’ and ‘h’ itself are the inbuilt input signals. ‘b’ is the given message bit of length 16-bits. Whereas ‘e’ is the encoded error output. Further more, ‘ee’ is

the correct encoded output. Coming to the decoding section, 'dd' is the decoded error output. 'd' is the correct output of the decoded operation. As the algorithm contains the half adder the results of the XOR operation of a and b is given as 'f' and the AND operation result is represented as't'.

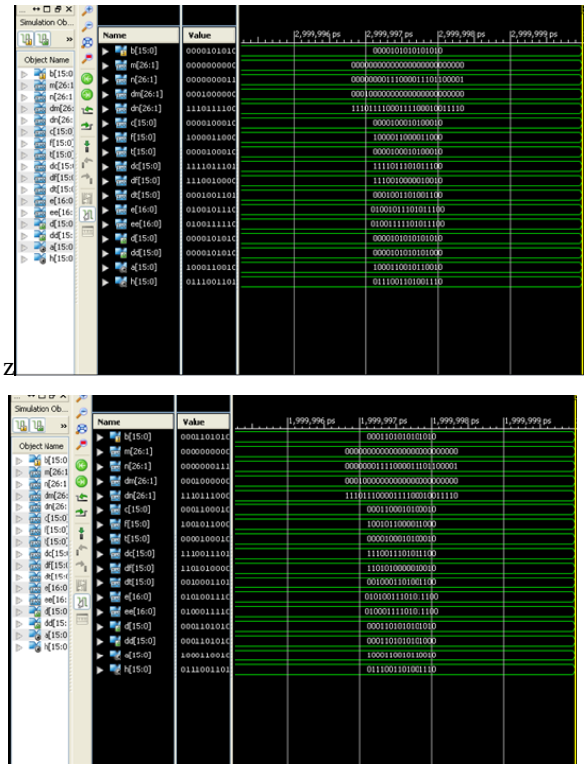

Fig.7. Simulated outputs for different inputs

For any given value of the input 'b' the decoded output'd' must be equal. Thus the transmitted message at the transmitter must be equal to the received message at the receiver i.e. the decoder output. From the above output results, the transmitted message is equal to the received message. Hence it is clear that both are identical. Hence the data is transmitted without any loss.

TABLE II: Comparison table

Architecture	No. of slices used	Delay (ns)	Power Consumption (m W)
Direct compare	785	43.766	1741
BWA	612	43.17	1259
Modified BWA	591	36.233	982

As appeared in table II, the proposed design is efficient in reducing the delay and also the power consumption. It is evident that the viability of the proposed design over the direct compare method available in literature like [6] is clearly mentioned in table II. In the comparison of different parameters used in the Direct Compare Architecture, BWA architecture and the modified BWA architectures have been presented. The delay and power consumption is less in Modified BWA as shown in the above table.

V. CONCLUSION

To reduce the latency and hardware complexity, a new architecture has been presented for matching the data protected with an ECC. The designed architecture examines whether the incoming data matches the cached data if a certain number of erroneous bits are corrected. To reduce the latency, the comparison of the data is parallelized with the encoding process and bubbled gates were used. Therefore, an efficient processing architecture has been presented to further minimize the latency and complexity.

VI. REFERENCES

[1] J. Chang, M. Huang, J. Shoemaker, J. Benoit, S.-L. Chen, W. Chen, Chiu, R. Ganesan, G. Leong, V. Lukka, S. Rusu, and D. Srivastava, "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, Apr. 2007.

[2] J. D. Warnock, Y.-H. Chan, S. M. Carey, H. Wen, P. J. Meaney, G. Ger-wig, H. H. Smith, Y. H. Chan, J. Davis, P. Bunce, A. Pelella, D. Rodko, P. Patel, T. Strach, D. Malone, F. Malgioglio, J. Neves, D. L. Rude, and W. V. Huott "Circuit and physical design implementation of the microprocessor chip for the zEnterprise system," *IEEE J. Solid-State Circuits*, *vol. 47*, *no. 1*, pp. 151–163, Jan. 2012.

[3] H. Ando, Y. Yoshida, A. Inoue, I. Sugiyama, T. Asakawa, K. Morita, T. Muta, and T. Motokurumada, S. Okada, H. Yamashita, and Y. Sat-sukawa, "A 1.3 GHz fifth generation SPARC64 microprocessor," in *IEEE ISSCC. Dig. Tech. Papers*, Feb. 2003, pp. 246–247.

[4] M. Tremblay and S. Chaudhry, "A third-generation 65nm 16-core 32-thread plus 32-scout-thread CMT SPARC processor," in *ISSCC. Dig. Tech. Papers*, Feb. 2008, pp. 82–83.

[5] AMD Inc. (2010). *Family 10h AMD Opteron Processor Product Data Sheet*, Sunnyvale, CA, USA [Online]. Available: http://support.amd.com/us/Processor_TechDocs/40036.pdf

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

[7] S. Lin and D. J. Costello, *Error Control Coding: Fundamentalsand Applications*, 2nd ed. Englewood Cliffs, NJ, USA: Prentice-Hall, 2004.

[8] Y. Lee, H. Yoo, and I.-C. Park, "6.4Gb/s multi-threaded BCH encoder and decoder for multi-channel SSD controllers," in ISSCC Dig. Tech. Papers, 2012, pp. 426–427

[9] Byeong Yong Kong,Jihyuck Jo,Hyewon Jeong,Mina Hwang,Soyoung Cha,Bongjin Kim,and In-Cheol Park, "Low – complexity Low-latency architecture for matching of data encoded with hard systematic error correcting codes".

Satyakiran Kambhampati Currently working as an assistant professor in the ECE department of GMR Institute of Technology, Rajam. His research interests include Low Power VLSI and Digital System Design.