IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 21, Issue 2, Ser. II (Mar - Apr 2019), PP 65-70
www.iosrjournals.org

Analyzing Contemporary Control Hazard Resolution Techniques

Amit Pandey[1], Abdella K. Mohammed[1]
[1]*Faculty of Informatics, Hawassa University, Ethiopia*
Corresponding Author: Amit Pandey

Abstract: *A basic RISC style pipelined processor can be divided in to five different stages, namely Instruction fetch (IF), Instruction Decode (ID), Execution (Ex), Memory (Mem) and Write back (WB). These stages are arranged to work in a synchronized way. With each clock pulse each stage forwards its instruction to the next consecutive stage. Initially each instruction will enter into the Instruction fetch stage. Further it will be escalated to the final Write back stage in same fashion. In the WB stage the outcome of the instruction will be stored in the general purpose register (GPR). Problem arises when there is a control instruction. Conceptually, the control instructions are evaluated in ID stage. However, by the time it gets evaluated the next instruction enters the IF stage. This can cause a control hazard. The current study consists analysis of various dynamic branch prediction techniques that are used to resolve the control hazard.*

Keywords: *Control hazard, Dynamic Branch Prediction, Branch Prediction, Pipeline hazards, Control Hazard resolution.*

Date of Submission: 27-03-2019 Date of acceptance: 11-04-2019

I. Introduction

The delay caused in the ID stage is solely responsible for introducing the control hazard. To handle these control hazards there are two types of strategies. First is Static Branch Prediction and second is Dynamic Branch Prediction. Static prediction techniques basically follow a fixed approach for every scenario and hence cannot exceed a certain level of performance threshold. On the other hand Dynamic predictions are based on the current scenario and hence they supersede the Static predictions in sense of performance. In the scope of this paper we will undergo some of the spectacularly performing Dynamic branch prediction techniques.

II. Dynamic Branch Prediction Techniques

2.1 G-Share Predictor

As tons of superfluousness exist in the counter index in the GSelect predictor, hence to take advantage from the hashing of global history collectively with the branch address McFarling proposed an alternative for the correlating predictor in 1993, acknowledged as GShare predictor. McFarling suggested that direct use of the global history is ineffective and its efficiency can be enhanced by implementing XOR of branch address with the global history [1] and using the above approach results even better than GSelect predictor were achieved [2] [3] (See Fig 1).

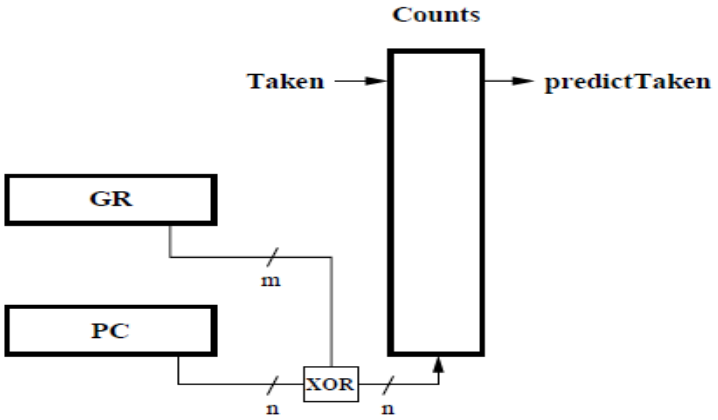

Fig 1: GShare Predictor

This marginally improved the efficiency of GShare [1], this can furthermore be seen in Fig 2.

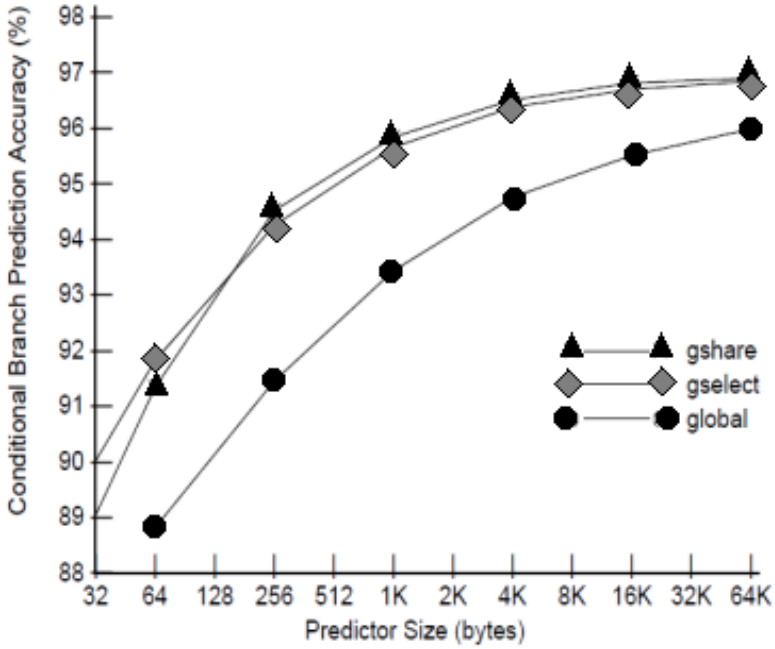

Fig 2: Performance Assessment of Global, GSelect and GShare Predictors

2.2 Tournament Predictor / Hybrid Predictor

Tournament or Hybrid predictor is merely a combination of more than one branch predictors. As various predictors may perform contrarily under varying branch conditions. Hence a blend of predictors is taken with a 2-bit selection counter [4] [5] [6] [7] (See Fig 3). Every predictor has its individual index value. When first predictor fails and second one holds the accurate result then the selection counter value is decreased and when the first predictor holds the accurate result and second flops then the selection counter value is increased [8] [9] [10] (See Table 1). See Fig 4, for the performance analysis of the Hybrid predictor.

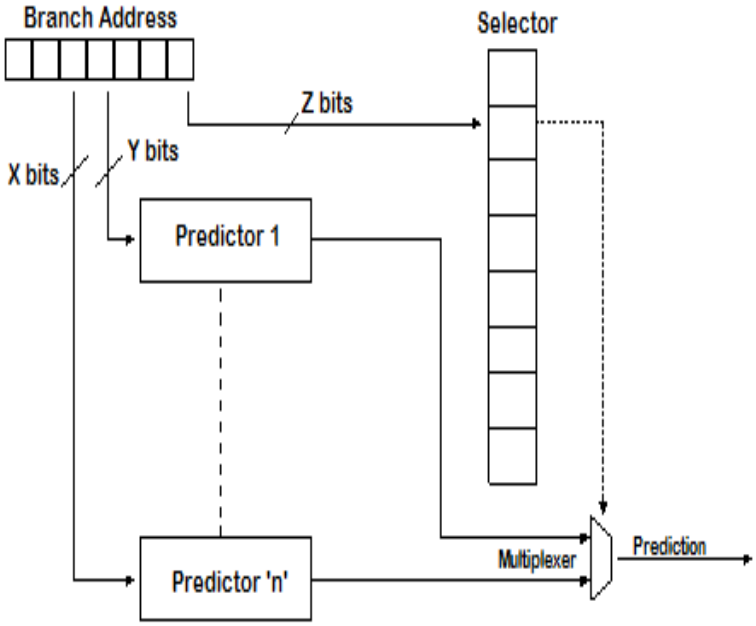

Fig 3: Hybrid Predictor / Tournament Predictor

Table 1: Selector Counter Update in Hybrid Predictor

	Predictor 1 Counter	Predictor 2 Counter	Increment / Decrement
Both Wrong	0	0	0 (No Change)
Predictor 1 Correct	0	1	-1 (Decrement)
Predictor 2 Correct	1	0	+1 (Increment)
Both Correct	1	1	0 (No Change)

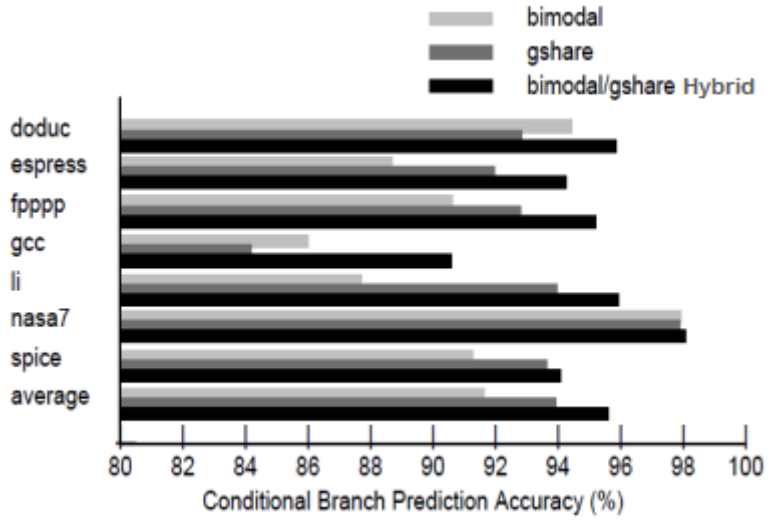

Fig 4: Performance of Bimodal / GShare Hybrid Predictor on SPEC 89 benchmarks

2.3 Fusion-Based Hybrid Predictors / Combined Output Lookup Table (COLT)

A Hybrid predictor always chooses one from the two or more participating predictors. By choosing the required one, the information contents of the other predictors are completely dumped. Taking this as point of improvement Loh and Henry in 2002, suggested a novel Fusion-Based Hybrid predictor COLT. Instead of choosing any particular predictor for making the absolute prediction, it uses an inter-mapped table lookups for devising the Fusion-Based Hybrid predictor. This Hybrid predictor that utilizes the information content related to all the other contributing predictors. With every effective guess, there is a lookup in the adjacent records of the mapping tables [11]. Suppose, there are three contributing predictors P1, P2, P3 and they predict the branches as 1, 0, 1 respectively. Now, the Fusion-Based Hybrid Predictor (COLT) lookup the mapping table, into each counter for the entry 101. If the records from 001 and 101 lead to dissimilar prediction. Then the prediction from P1 will be considered for deciding the ultimate result. However, if both the records lead to same prediction, then prediction from P1 is not a decisive element in the mapping. Furthermore, when the counter in a contiguous record is not appropriately trained, then this strategy may work faulty (See Fig 5).

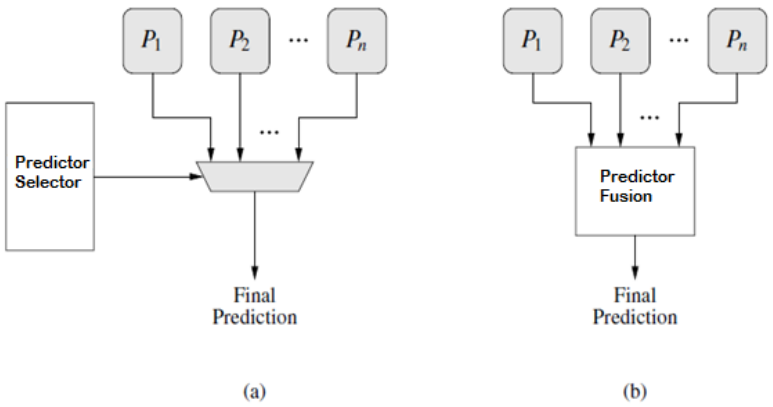

Fig 5: a) Usual Hybrid Predictor**.** **b)** Fusion-Based Hybrid Predictor

2.4 Neural Methods for Dynamic Branch Prediction

Perceptrons were initially presented in 1961 to learn brain functions [12]. The primitive solitary layer perceptron comprise of single artificial neuron. It allocates weights that are evaluated using certain algorithm, on its inputs to render a final result [13]. A perceptron is taught using a training function of n inputs. Typically, these inputs are bits from global branch history shift register and the function chooses whether the present branch will be taken or rejected. Perceptron is signified as a vector whose constituents are weights or signed integers. Now, the preliminary weight element is termed as 'biase' and its value is permanently '1' (See Fig 6). The final result is calculated as dot product of the weight vector and the input vector, as displayed below.

$$Perceptron\ Output = W_0 + \sum_{j=1}^{n} (X_j W_j)$$

Now inputs to the perceptron will take only two values. '-1' if the branch is not taken and '+1' if the branch is taken. Likewise, a negative output refers to predict branch not taken and a non-negative output refers to predict branch taken.

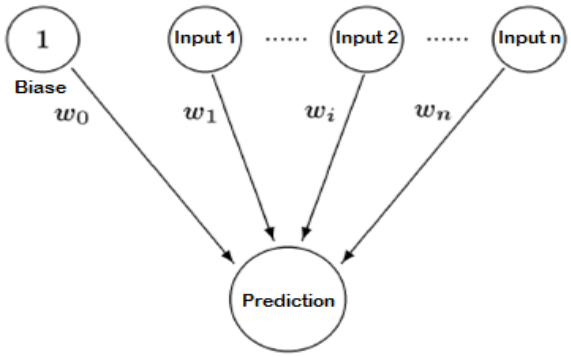

Fig 6: Single layer Perceptron Model

Every weight demonstrations the level of association between the activities of a preceding branch and the nature of the present branch. Non-negative weights demonstrates positive correlation, and negative weights demonstrates -ive correlation. After the branch is engaged then the corresponding weight is added, otherwise the weight is deducted. If the resultant sum is non-negative then the branch is predict taken, otherwise it will be overruled (See Fig 7). The perceptron predictor that serves the global information, has 14 % better rate of prediction than the McFarling type hybrid predictor [14].

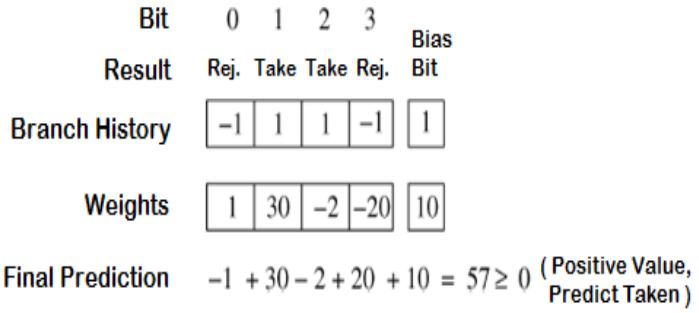

Fig 7: Perceptron prediction mechanism

In 1997, Calder et al. suggested a neural network based static branch prediction technique [15]. The technique uses information such as opcode and control flow as input to a neural network, for predicting the branch direction during compile time. But with a misprediction percentage of 20% still this method was superior to former static branch prediction methods with misprediction percentage up to 25% [16]. But still when compared to existing dynamic prediction techniques its performance was not up to mark. Emer and Gloy in 1997, also suggested a branch predictor based on genetic algorithm [17].

In 1999, Lucian et al. has suggested the first dynamic branch prediction algorithm based on neural network [18]. It was based on the concept of how neurons learns vector quantization. The precision of this algorithm was analogous to that of history table based branch predictors [11] [19]. These predictors were also used as constituent, for study purposes, in Hybrid predictors. Likewise, it is the utmost accurate branch predictor with single constituent in documents [11] [14]. While in theory these predictors accomplish better performance

than hybrid predictors. But in practical, due to their fetch latency their efficiency is not up to mark. Jiménez et al. in 2000 has analyzed the techniques for justifying the latency in branch predictors [20]. Typically, a quick but reasonably imprecise predictor advances to lead the branch prediction. However it can be enhanced by selecting a slower but more accurate predictor over the earlier. This recurrently used technique is known as overriding. This technique was used in the Alpha EV6 and EV7 cores [21] and was also projected for the Alpha EV8 [22].

Jiménez in 2003 suggested a novel branch predictor based on neural network. To diminish the fetch latency the suggested method chooses its weight vector depending on the path taking to branch, instead of considering the branch address only [23]. This tactic has two prominent qualities. First, the fetch delay is diminished. Second, the accuracy is boosted as the predictor also receives the path information for the forecast (See Fig 8).

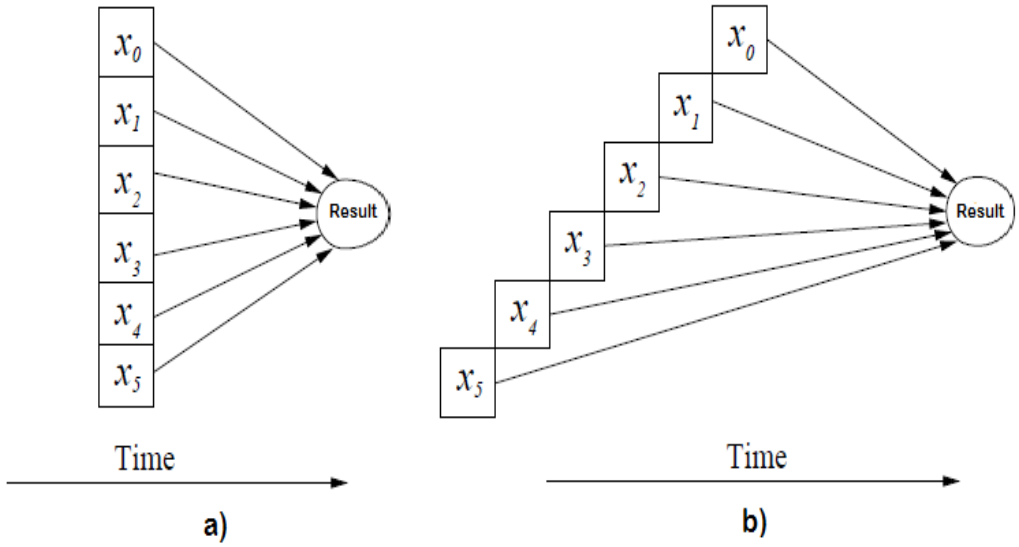

Fig 8: a) Usual perceptron based predictor. **b)** Path based neural branch predictor

Further, Monchiero & Palermo in 2005 suggested a method that uses two different perceptron based predictors employed concurrently to construe the final result [24]. This method uses address based perceptron predictor together with history based perceptron predictors for making the final prediction (See Fig 9).

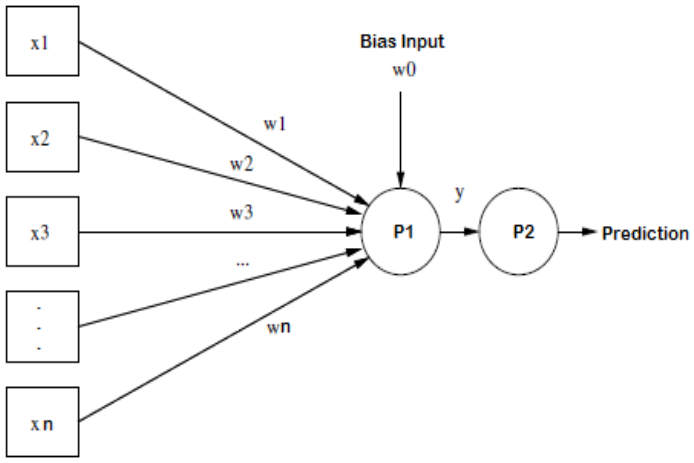

Fig 9: Combined Perceptron based branch predictor

III. Conclusion

This study reflected that the efficiency of McFarling's G-Share predictor is marginally better than the G-Select predictor. Later, the Hybrid predictors has shown efficiency even better than that of G-Share's.

Further, the perceptron predictor that serves the global information, has 14 % better rate of prediction than the McFarling type hybrid predictor.

The neural network based static branch prediction technique suggested by Calder et al. was superior to former static branch prediction methods, but still when compared to existing dynamic prediction techniques its performance was not up to mark.

Lucian et al. has suggested the first neural network based dynamic branch prediction algorithm. While in theory these predictors accomplish better performance than hybrid predictors. But in practical, due to their fetch latency their efficiency is not up to mark.

References

[1]. S. McFarling, *Combining branch predictors*. Technical Report TN-36, Digital Western Research Laboratory, Vol. 49,1993.
[2]. C. C. Lee, I. C. K. Chen, and T. N. Mudge, The bi-mode branch predictor. *In Proceedings of the 30th annual ACM/IEEE international symposium on Microarchitecture*. 1997, 4-13.
[3]. M. Evers, S. J. Patel, R.S. Chappell, & Y. N. Patt, *An analysis of correlation and predictability: What makes two-level branch predictors work*. In ACM SIGARCH Computer Architecture News. Vol. 26, No. 3, 1998, 52-61.
[4]. M. Evers, & T. Y. Yeh, Understanding branches and designing branch predictors for high-performance microprocessors. *Proceedings of the IEEE*, 89(11), 2001, 1610-1620.
[5]. A. Falcon, J. Stark, A. Ramirez, K. Lai & M. Valero, *Prophet/critic hybrid branch prediction*. In ACM SIGARCH Computer Architecture News. Vol. 32, No. 2, 2004, p. 250
[6]. A. Seznec & P. Michaud, *De-aliased hybrid branch predictors*, Doctoral dissertation, INRIA, 1999.
[7]. K. Skadron, M. Martonosi & D. W. Clark, A taxonomy of branch mispredictions, and alloyed prediction as a robust solution to wrong-history mispredictions. *In Parallel Architectures and Compilation Techniques, Proceedings of IEEE International Conference on*. 2000, 199-206.
[8]. P. Y . Chang, E. Hao & Y. N. Patt, Alternative implementations of hybrid branch predictors. *In Microarchitecture, Proceedings of the 28th Annual IEEE International Symposium on.* 1995, 252-257.
[9]. M. Evers, P. Y. Chang & Y. N. Patt, *Using hybrid branch predictors to improve branch prediction accuracy in the presence of context switches.* In ACM SIGARCH Computer Architecture News, Vol. 24, No. 2, 1996, 3-11.
[10]. D. Grunwald, D. Lindsay & B. Zorn, Static methods in hybrid branch prediction. *In Parallel Architectures and Compilation Techniques, Proceedings of IEEE Conference on*, 1998, 222-229.
[11]. G. H. Loh & D. S. Henry, Predicting conditional branches with fusion-based hybrid predictors. *In Parallel Architectures and Compilation Techniques, Proceedings of IEEE International Conference on*. 2002, 165-176.
[12]. F. Rosenblatt, Principles of neurodynamics. perceptrons and the theory of brain mechanisms. *CORNELL AERONAUTICAL LAB INC BUFFALO NY.* No. VG-1196-G-8, 1961.
[13]. H. D. Block, *The perceptron: A model for brain functioning*. Reviews of Modern Physics, 34(1), 1962, 123.
[14]. D. A. Jiménez & C. Lin, *Neural methods for dynamic branch prediction*. ACM Transactions on Computer Systems (TOCS), 20(4), 2002, 369-397.
[15]. B. Calder, D. Grunwald, M. Jones, D. Lindsay, J. Martin, M. Mozer & B. Zorn, *Evidence-based static branch prediction using machine learning*. ACM Transactions on Programming Languages and Systems (TOPLAS), 19(1), 1997, 188-222.
[16]. T. Ball & J. R. Larus, *Branch prediction for free,* ACM Vol. 28, No. 6, 1993, 300-313.
[17]. J. Emer & N. Gloy, *A language for describing predictors and its application to automatic synthesis*. In ACM SIGARCH Computer Architecture News, Vol. 25, No. 2, 1997, 304-314.
[18]. L. N. Vintan & M. Iridon, Towards a high performance neural branch predictor. *Neural Networks, IJCNN'99, In proceedings of IEEE International Joint Conference on. Vol. 2*, 1999, 868-873.
[19]. R. Thomas, M. Franklin, C. Wilkerson & J. Stark, *Improving branch prediction by dynamic dataflow-based identification of correlated branches from a large global history*. In ACM SIGARCH Computer Architecture News, Vol. 31, No. 2, 2003, 314-323.
[20]. D. A. Jiménez, S. W. Keckler & C. Lin, The impact of delay on the design of branch predictors. *In Microarchitecture, MICRO-33. In Proceedings of 33rd Annual IEEE/ACM International Symposium on.* 2000, 67-76.
[21]. R. E. Kessler, *The alpha 21264 microprocessor*. IEEE micro, 19(2), 1999, 24-36.
[22]. A. Seznec, S. Felix, V. Krishnan & Y. Sazeides, Design tradeoffs for the Alpha EV8 conditional branch predictor. *Computer Architecture, In Proceedings of IEEE 29th Annual International Symposium on*. 2002, 295-306.
[23]. D. A. Jiménez, Fast path-based neural branch prediction. *In Proceedings of the 36th annual IEEE/ACM International Symposium on Microarchitecture.* 2003, 243.
[24]. M. Monchiero & G. Palermo, The combined perceptron branch predictor. *In European Conference on Parallel Processing, Springer*. 2005, 487-496.

IOSR Journal of Computer Engineering (IOSR-JCE) is UGC approved Journal with Sl. No. 5019, Journal no. 49102.

Amit Pandey. "Analyzing Contemporary Control Hazard Resolution Techniques" IOSR Journal of Computer Engineering (IOSR-JCE) 21.2 (2019): 65-70.