International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

Comparative Analysis of Branch Prediction Techniques Across Diverse Benchmark Suites

Sai Kumar Marri[1], E. Sikender[2]

[1]Student Researcher, Department of Electrical Engineering, UTD
[2]Student Researcher, Department of Electrical and Communications Engineering, VCE

Abstract

Branch prediction is a critical aspect of modern microprocessor design, significantly influencing performance and energy efficiency in pipelined architectures. Accurate branch predictors reduce pipeline stalls, enhance instruction-level parallelism and overall system throughput. This study provides a comprehensive analysis of various branch prediction techniques, including bimodal predictors, perceptron-based predictors, hybrid schemes, and low-power alternatives, as applied to diverse benchmark suites such as SPEC CPU2000, Mibench, and Mediabench. The paper explores the architectural principles, advantages, and limitations of these predictors, emphasizing their accuracy, power consumption, and hardware overhead. Key innovations like genetic algorithm-enhanced predictors and neural network-based designs are discussed, highlighting their ability to adapt dynamically to workload characteristics [6][2]. Furthermore, the study examines novel approaches, such as undervolting predictors for energy efficiency [5] and complementary predictors designed to address misprediction patterns [4].

Simulation results obtained from platforms like SimpleScalar and gem5 demonstrate the predictors' performance across various benchmarks, revealing trade-offs between prediction accuracy, computational complexity, and energy efficiency. For instance, perception-based predictors show superior accuracy with long history lengths [2][6], whereas low-power designs like TBIT minimize energy usage with negligible performance degradation, making them ideal for embedded systems [7]. Hybrid predictors, which combine global and local history, strike a balance between performance and complexity [3][6]. This analysis highlights the evolving landscape of branch prediction technologies and underscores the importance of tailoring predictor designs to specific application domains. By leveraging these insights, designers can optimize processors for high performance, reduced power consumption, or a combination of both, meeting the demands of modern computing systems. This work serves as a foundational reference for advancing branch predictor research and development in future architectures [1][5].

Keywords: Branch Prediction, Processor Performance, Energy Efficiency, Benchmark Analysis, Hybrid Predictors.

1. Introduction

Branch prediction plays a pivotal role in modern microprocessor architecture, serving as a cornerstone for achieving high performance and energy efficiency in pipelined systems. The increasing complexity of applications and the demand for faster processing speeds have driven the development of

sophisticated branch predictors capable of minimizing control hazards. By speculatively predicting the outcome of branch instructions, these predictors enable the processor to maintain an uninterrupted instruction flow, thereby reducing pipeline stalls and maximizing instruction-level parallelism (ILP) [1][3]. Branch prediction accuracy is crucial to system performance, as mispredictions lead to pipeline flushing, wasted computational resources, and additional energy consumption. The evolution of branch prediction strategies has introduced a variety of designs, ranging from simple static and dynamic predictors to more complex neural network-based and hybrid approaches. Each of these designs offers unique trade-offs in terms of prediction accuracy, power consumption, and implementation complexity, making it essential to evaluate their performance under varying workloads and application domains [2][6][7].

This paper presents a comprehensive analysis of several branch prediction techniques, focusing on their implementation and performance across diverse benchmark suites such as SPEC CPU2000, Mibench, and Mediabench. The study includes traditional methods like bimodal and global history-based predictors, as well as advanced techniques such as perception-based predictors, hybrid predictors combining local and global histories, and low-power designs tailored for embedded systems [3][7]. Additionally, novel approaches like genetic algorithm-optimized predictors and undervolting strategies are explored, highlighting their potential to enhance energy efficiency without compromising performance [5][6]. The benchmarks selected for this analysis represent a broad spectrum of real-world applications, enabling a thorough evaluation of predictor performance across computationally intensive tasks and power-sensitive scenarios. Simulation tools such as SimpleScalar and gem5 provide a detailed modeling environment, facilitating the measurement of prediction accuracy, misprediction rates, energy consumption, and hardware overhead [2][6].

The objective of this study is to provide valuable insights into the strengths and limitations of different branch prediction techniques, guiding processor designers in selecting or developing optimal predictors for specific applications. As modern systems increasingly prioritize a balance between performance and energy efficiency, understanding these trade-offs becomes critical. By comparing various predictors under standardized benchmarks, this work aims to contribute to the ongoing advancements in microprocessor architecture, supporting the development of more efficient and adaptive computing systems [1][4]. In the subsequent sections, we delve into the architectural details of each predictor type, analyze their simulation results, and discuss their suitability for different computing environments, paving the way for future innovations in branch prediction technology.

2. Setting up the Simulator

For the analysis of the branch predictors, we are required to compile the simulator enabling different branch predictors. We worked with the source files from the gem5.org website but the latest simulator versions have issues in reporting the number of BTB hits (BTBHits). This parameter always reported as zero and eventually BTBMisPct reported as 100% for all the predictors. By default, the branch predictor support not enabled in the simulator. We enabled the branch predictor by modifying BaseSimpleCPU.py. Different branch predictors enabled and recompiled the simulator. Simulator is compiled using the command scons ./build/X86/gem5.opt. After recompilation, simulator sanity is verified by running the simple hello world program on the simulator to cross check for errors.

Simulator compilation and running a simple hello world program generates output files in m5out directory. Posting the config.ini results from the three branch predictors. We have added additional

parameters BTBMissPct and BranchMispredPercent to the source files and generated the stats file with the new parameters.

```
class BaseSimpleCPU(BaseCPU):
    type = 'BaseSimpleCPU'
    abstract = True
    cxx_header = "cpu/simple/base.hh"

    def addCheckerCpu(self):
        if buildEnv['TARGET_ISA'] in ['arm']:
            from ArmTLB import ArmTLB

            self.checker = DummyChecker(workload = self.workload)
            self.checker.itb = ArmTLB(size = self.itb.size)
            self.checker.dtb = ArmTLB(size = self.dtb.size)
        else:
            print "ERROR: Checker only supported under ARM ISA!"
            exit(1)

    branchPred = Param.BranchPredictor(TournamentBP(), "Branch Predictor")
    #branchPred = Param.BranchPredictor(BiModeBP(), "Branch Predictor")
    #branchPred = Param.BranchPredictor(LocalBP(), "Branch Predictor")
```

Figure 1: Adding support for branch predictor to simulator

a. BTBMisPct: This parameter computes the BTB miss percentage. We have used existing parameters to derive the miss percentage.

BTBMisPct = (1 – (BTBHits/BTBLookups)) * 100

BTBHits – number of BTB hits

BTBLookups – number of BTB references

b. BranchMispredPercent: This parameter computes the percentage of branch misprediction.

BranchmispredPercent = (numBranchMispred / numBranches) * 100

numBranchMispred – number of mispredicted branches

numBranches – number of branches fetched

```
103
104     BTBHitPct
105         .name(name() + ".BTBHitPct")
106         .desc("BTB Hit Percentage")
107         .precision(6);
108     BTBHitPct = (BTBHits / BTBLookups) * 100;
109
110
111     BTBMissPct
112         .name(name() + ".BTBMissPct")
113         .desc("BTB Miss Percentage")
114         .precision(6);
115     BTBMissPct = (1 - (BTBHits / BTBLookups)) * 100;
116
117
```

Figure 2: source code changes (bpred_unit.cc)

To add support for BTBMisPct, we modified bpred_unit.cc and bpred_unit.hh source directory - gem5/src/cpu/pred

```
363
364         t_info.idleFraction = constant(1.0) - t_info.notIdleFraction;
365         t_info.numIdleCycles = t_info.idleFraction * numCycles;
366         t_info.numBusyCycles = t_info.notIdleFraction * numCycles;
367         t_info.BranchMispredPercent = (t_info.numBranchMispred / t_info.numBranches) * 100;
368
369         t_info.numBranches
370             .name(thread_str + ".Branches")
371             .desc("Number of branches fetched")
372             .prereq(t_info.numBranches);
373
374         t_info.numPredictedBranches
375             .name(thread_str + ".predictedBranches")
376             .desc("Number of branches predicted as taken")
377             .prereq(t_info.numPredictedBranches);
378
379         t_info.numBranchMispred
380             .name(thread_str + ".BranchMispred")
381             .desc("Number of branch mispredictions")
382             .prereq(t_info.numBranchMispred);
383
384
385         t_info.BranchMispredPercent
386             .name(thread_str + ".BranchMispredPercent")
387             .desc("Number of branch mispredictions percentage");
```

Figure 3: Source code changes (base.cc)

3. Analyzing Benchmarks with Different Branch Predictors

We have recompiled the simulator after adding the additional parameters to the source code and checked the parameter (BTBMisPct and BranchMispredPercent) numbers for different branch predictors. These numbers are generated for simple hello world programs and compared the results.

Table 1: Analysis of different branch predictors on hello world program

Parameters	TournamentBP	BiModeBP	LocalBP
BTBLookups (# BTB lookups)	998	203	435
BTBHits (# BTB hits)	367	202	368
BTBHitPct (BTB hit percentage)	36.773%	99.507%	84.597%
BTBMissPct (BTB miss percentage)	**63.226%**	**0.492%**	**15.402%**
Branches (# branches fetched)	1317	1317	1317
predictedBranches (# branches predicted as taken)	472	210	405
BranchMispred (# branch mispredictions)	477	621	468
BranchMispredPercent (miss prediction percentage)	**36.218%**	**47.152%**	**35.535%**

From the analysis of three branch predictors:

- BTB hit percentage is maximum with the BiMode branch predictor and minimum with the Tournament branch predictor.
- Branch misprediction percentage is maximum with the BiMode branch predictor and almost same misprediction percentage with both LocalBP and TournamentBP.

We compiled the simulator with different branch predictors and simulated five benchmarks. These benchmarks compare all the three branch predictors to understand the effectiveness of each branch predictors. These benchmarks are run up to five million instructions.

a. Benchmark-I (401.bzip2)

Parameters (401.bzip2 benchmark)	TournamentBP	BiModeBP	LocalBP
BTBLookups (# BTB lookups)	33231142	33071034	33189670
BTBHits (# BTB hits)	33196123	33057641	33159138
BTBHitPct (BTB hit percentage)	99.895%	99.96%	99.909%
BTBMissPct (BTB miss percentage)	**0.105%**	**0.040%**	**0.091%**
Branches (# branches fetched)	37843295	37843292	37843295
predictedBranches (# branches predicted as taken)	33922687	33784124	33885896
BranchMispred (# branch mispredictions)	2274103	2243189	2490136
BranchMispredPercent (miss prediction percentage)	**6.009%**	**5.927%**	**6.580%**
CPI	**1.0882**	**1.0882**	**1.0882**

- Benchmark – I have a better branch prediction with the BiMode branch predictor.
- Benchmark – I have a better BTB hit percentage with the BiMode branch predictor.
- All the branch predictors have resulted similar CPI with the benchmark – I.

b. Benchmark-II (429.mcf)

Parameters (429.mcf benchmark)	TournamentBP	BiModeBP	LocalBP
BTBLookups (# BTB lookups)	51478564	49457318	52004940
BTBHits (# BTB hits)	49259638	49171235	50467083
BTBHitPct (BTB hit percentage)	95.689%	99.421%	97.042%
BTBMissPct (BTB miss percentage)	**4.310%**	**0.578%**	**2.957%**
Branches (# branches fetched)	97901000	97901000	97901001
predictedBranches (# branches predicted as taken)	54759994	54679836	55979123
BranchMispred (# branch mispredictions)	4522359	5177396	9538849
BranchMispredPercent (miss prediction percentage)	**4.619%**	**5.288%**	**9.743%**
CPI	**1.7485**	**1.7485**	**1.7485**

- Benchmark – II has a better branch prediction with the Tournament branch predictor.
- Benchmark – II has a better BTB hit percentage with the BiMode branch predictor.
- All the branch predictors have resulted similar CPI with the benchmark – II.

c. Benchmark-III (456.hmmer)

Parameters (456.hmmer benchmark)	TournamentBP	BiModeBP	LocalBP
BTBLookups (# BTB lookups)	20472409	20162781	20980599
BTBHits (# BTB hits)	20236959	20137783	20639723
BTBHitPct (BTB hit percentage)	98.85%	99.876%	98.375%
BTBMissPct (BTB miss percentage)	**1.15%**	**0.123%**	**1.624%**
Branches (# branches fetched)	27593152	27593147	27593152
predictedBranches (# branches predicted as taken)	21002432	20902613	21404838
BranchMispred (# branch mispredictions)	2215893	2787987	3947484
BranchMispredPercent (miss prediction percentage)	**8.03%**	**10.103%**	**14.306%**
CPI	**1.002**	**1.002**	**1.002**

- Benchmark – III has a better branch prediction with the Tournament branch predictor.
- Benchmark – III has a better BTB hit percentage with the BiMode branch predictor.
- All the branch predictors have resulted similar CPI with the benchmark – III.

d. Benchmark- IV (458.sjeng)

Parameters (458.sjeng benchmark)	TournamentBP	BiModeBP	LocalBP
BTBLookups (# BTB lookups)	53090884	52487285	55687877
BTBHits (# BTB hits)	51490367	51073624	52296300
BTBHitPct (BTB hit percentage)	96.985%	97.306%	93.909%
BTBMissPct (BTB miss percentage)	**3.014%**	**2.693%**	**6.090%**
Branches (# branches fetched)	69678398	69678397	69678398
predictedBranches (# branches predicted as taken)	54291570	53882697	55118774
BranchMispred (# branch mispredictions)	5764538	5380561	8939533
BranchMispredPercent (miss prediction percentage)	**8.273%**	**7.721%**	**12.829%**
CPI	**1.9386**	**1.9386**	**1.9386**

- Benchmark – IV has a better branch prediction with the BiMode branch predictor.
- Benchmark – IV has a better BTB hit percentage with the BiMode branch predictor.
- All the branch predictors have resulted similar CPI with the benchmark – IV.

e. Benchmark- V (470.lbm)

Parameters (470.lbm benchmark)	TournamentBP	BiModeBP	LocalBP
BTBLookups (# BTB lookups)	15273542	8164887	9348818
BTBHits (# BTB hits)	8167849	8164786	8164290
BTBHitPct (BTB hit percentage)	53.477%	99.99%	87.329%
BTBMissPct (BTB miss percentage)	**46.522%**	**0.00123%**	**12.67%**
Branches (# branches fetched)	18089548	18089547	18089549
predictedBranches (# branches predicted as taken)	9352511	9349229	9348845
BranchMispred (# branch mispredictions)	64868	64452	107672
BranchMispredPercent (miss prediction percentage)	**0.358%**	**0.356%**	**0.595%**
CPI	**1.8**	**1.8**	**1.8**

- Benchmark – V has a better branch prediction with the BiMode branch predictor.
- Benchmark – V has a better BTB hit percentage with the BiMode branch predictor.
- All the branch predictors have resulted similar CPI with the benchmark – V.

4. Modifying the Branch Predictor sizes

For this analysis, we choose tournament branch predictor, modified the sizes of the predictor. Gem5 simulator recompiled with new predictor sizes and simulated for all the benchmarks. Predictor sizes were modified in this file: gem5/src/cpu/pred/BranchPredictor.py

a. Benchmark – I (401.bzip2)

Branch Parameters (Benchmark – I (401.bzip2))	Default settings	Modified
BTBLookups (# BTB lookups)	33231142	33243014
BTBHits (# BTB hits)	33196123	33196274
BTBHitPct (BTB hit percentage)	99.894%	99.859%
BTBMissPct (BTB miss percentage)	**0.105%**	**0.140%**
Branches (# branches fetched)	37843295	37843296
predictedBranches (# branches predicted as taken)	33922687	33922799
BranchMispred (# branch mispredictions)	2274103	2298269
BranchMispredPercent (miss prediction percentage)	**6.009%**	**6.073%**
CPI	**1.0882**	**1.088**

With the modified predictor sizes, this benchmark has minimal decrease in the branch prediction BTB hit percentage. New branch predictor numbers were not effective with this benchmark.

b. Benchmark – II (429.mcf)

Branch Parameters	Default settings	Modified
BTBLookups (# BTB lookups)	51478564	51424871
BTBHits (# BTB hits)	49259638	49039676
BTBHitPct (BTB hit percentage)	95.689%	95.361%
BTBMissPct (BTB miss percentage)	**4.310%**	**4.638%**
Branches (# branches fetched)	97901000	97901000
predictedBranches (# branches predicted as taken)	54759994	54540780
BranchMispred (# branch mispredictions)	4522359	4866788
BranchMispredPercent (miss prediction percentage)	**4.619%**	**4.971%**
CPI	**1.7485**	**1.7485**

With the modified predictor sizes, this benchmark has minimal decrease in the branch prediction BTB hit percentage. New branch predictor numbers were not effective with this benchmark.

c. Benchmark – III (456.hmmer)

Branch Parameters	Default settings	Modified
BTBLookups (# BTB lookups)	20472409	20570574
BTBHits (# BTB hits)	20236959	20375819
BTBHitPct (BTB hit percentage)	98.85%	99.053%
BTBMissPct (BTB miss percentage)	**1.15%**	**0.946%**
Branches (# branches fetched)	27593152	27593153
predictedBranches (# branches predicted as taken)	21002432	21141382
BranchMispred (# branch mispredictions)	2215893	2580723
BranchMispredPercent (miss prediction percentage)	**8.03%**	**9.352%**
CPI	**1.002**	**1.002**

With the modified predictor sizes, this benchmark has minimal improvement in the BTB hit percentage and minimal decrease in the branch prediction percentage.

d. Benchmark – IV (4578.sjeng2)

Branch Parameters	Default settings	Modified
BTBLookups (# BTB lookups)	53090884	53582018
BTBHits (# BTB hits)	51490367	51022784
BTBHitPct (BTB hit percentage)	96.985%	95.223707
BTBMissPct (BTB miss percentage)	**3.014%**	**4.776293**
Branches (# branches fetched)	69678398	69678398
predictedBranches (# branches predicted as taken)	54291570	53824513
BranchMispred (# branch mispredictions)	5764538	6824514
BranchMispredPercent (miss prediction percentage)	**8.273%**	**9.794304**
CPI	**1.9386**	**1.9386**

With the modified predictor sizes, this benchmark has minimal decrease in the branch prediction BTB hit percentage. New branch predictor numbers were not effective with this benchmark.

e. Benchmark – V (470.lbm)

Branch Parameters (Benchmark – V (470.lbm))	Default settings	Modified
BTBLookups (# BTB lookups)	15273542	15292438
BTBHits (# BTB hits)	8167849	8186736
BTBHitPct (BTB hit percentage)	53.477%	53.534%
BTBMissPct (BTB miss percentage)	**46.522%**	**46.465%**
Branches (# branches fetched)	18089548	18089547
predictedBranches (# branches predicted as taken)	9352511	9371397
BranchMispred (# branch mispredictions)	64868	117217
BranchMispredPercent (miss prediction percentage)	**0.358%**	**0.647%**
CPI	**1.8**	**1.8**

With the modified predictor sizes, this benchmark has minimal improvement in the BTB hit percentage and minimal decrease in the branch prediction percentage.

5. Conclusion

Branch prediction remains a cornerstone in advancing microprocessor performance and energy efficiency, with its accuracy directly impacting instruction-level parallelism and resource utilization. This study has explored a wide range of branch prediction techniques, from traditional approaches like bimodal and global history predictors to advanced perceptron-based and hybrid schemes. Through simulations across benchmark suites like SPEC CPU2000 and Mibench, we highlighted the trade-offs between prediction accuracy, power consumption, and hardware complexity inherent in these methods [1][3][6]. Innovative approaches, such as genetic algorithm-optimized predictors and undervolting strategies, demonstrate significant potential for improving efficiency without sacrificing performance [5][6]. While perceptron-based predictors excel in leveraging long history lengths for higher accuracy, hybrid designs strike a critical balance between simplicity and performance. Additionally, low-power predictors tailored for embedded systems effectively address energy constraints, making them ideal for mobile and IoT devices [2][7]. This comparative analysis underscores the importance of selecting prediction techniques aligned with specific application needs. As workloads grow more diverse, branch predictor designs must continue evolving, incorporating adaptive and energy-efficient strategies. This work serves as a foundation for future research, aiming to optimize the interplay between performance, power, and complexity in branch prediction systems [4][5].

6. References

1. Impact of Inaccurate Design of Branch Predictors on Processors' Power Consumption Baisakhi Das, Gunjan Bhattacharya, Ilora Maity, Biplab K Sikdar. IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing, 2011. DOI: 10.1109/DASC.2011.73.
2. A Study of Perceptron Based Branch Prediction on Simplescalar Platform Yang Lu, Yi Liu, He Wang. IEEE, 2011. DOI: Not available in provided snippet.
3. A Study for Branch Predictors to Alleviate the Aliasing Problem in Pipelining Tieling Xie, Robert Evans, Yul Chu. IEEE, 2005. DOI: 10.1109/ICCD.2005.603.

4. Branch Misprediction Prediction: Complementary Branch Predictors Resit Sendag, Joshua J. Yi, Peng-fei Chuang. IEEE Computer Architecture Letters, 2007. DOI: 10.1109/MCAL.2007.103.
5. Analysis and Characterization of Ultra Low Power Branch Predictors Athanasios Chatzidimitriou, George Papadimitriou, Dimitris Gizopoulos, Shrikanth Ganapathy, John Kalamatianos. IEEE International Conference on Computer Design, 2018. DOI: 10.1109/ICCD.2018.00030.
6. Enhancing Branch Predictors using Genetic Algorithm Md Sarwar M Haque, Salami Onoruoiza, Md Rafiul Hassan, Joarder Kamruzzaman, Muhammad Sulaiman, Md Arifuzzaman. 8th International Conference on Modeling Simulation and Applied Optimization (ICMSAO), 2019. DOI: 10.1109/ICMSAO.2019.00030.
7. Low Power Branch Predictor for Embedded Processors Sunwook Kim, Eutteum Jo, Hyungshin Kimi. IEEE International Conference on Computer and Information Technology (CIT), 2010. DOI: 10.1109/CIT.2010.59.