American Scientific Research Journal for Engineering, Technology, and Sciences (ASRJETS)

ISSN (Print) 2313-4410, ISSN (Online) 2313-4402

© Global Society of Scientific Research and Researchers

http://asrjetsjournal.org/

Implementation of Global History Branch Prediction Using MicroBlaze Processor

Raneen Alaa Ogla[a*], Safaa Sahab Omran[b]

[a,b]*Electrical Engineering Technical College, Middle Technical University, Baghdad, Iraq*

[a]*Email: Raneenalaa9@gmail.com*

[b]*Email: Omran_safaa@ymail.com*

Abstract

In this paper, using VHDL (Very high speed IC Hardware Description Language) hardware modeling the complete design of a 32-bit MIPS (Microprocessor without Interlocked Pipeline Stages) MicroBlaze processor is presented. The MIPS and top level is designed using (Xilinx vivado Design Suite 2018.1) program. Dynamic branch predictors are common because it can be achieve accurate results in branch prediction without change sequence execution instruction, in order to improve accuracy in branch desired, proposed new active (design) conditional branch predictor by connected with MicroBlaze processor and select number of entry that appropriate for these techniques by using bimodal dynamic branch predictor. A procedure that performs bimodal executed and the results are discussed this technique provides better prediction accuracy but require more power and complexity increases exponentially in design.

Keywords: Pattern History Table (PHT); Branch Predictor (BP); MicroBlaze (MB); Branch Target Address (BTA).

1. Introduction

Pipelining basically allow to interference execution of multiple operation at the same time, new processors used pipelining techniques to increase its performance. Pipelining performance is better when the execution of the orders is consecutive, but Pipeline performance changes if the instruction flow is changes, several types of hazards effected on it and cause pipeline stalls, and thus, it becomes important to propose ways to deal with the branches. The key idea behind pipelining is to divide the work into smaller pieces and use assembly line processing to complete the work [1,2]. Branches effect on control flow of instructions execution in irregular manner that interrupts the sequence fetch.

----

* Corresponding author.

To resume Instruction fetch must be taken in consideration the number of clock cycles that used until reachable the results of branches [3]. Branches are repeated in general purpose programs about 20% of instructions are used in branches. To reduce the delay of instructions methods It was used to predict the direction of the branch before the branch is selected, if the prediction is the correct track continued to fetch another instructions or divided executed instructions when predict is wrong [3]. Branch prediction is usually used to deal with a branch problem to improve performance by guessing the branch target to allow the processor to fetch the corresponding instructions and execute them speculatively. Support for more account-driven applications one of the key performance improvement methods is sub-prediction. Without branch prediction, the branch must be executed before the wizard can extract the following instructions. Prediction Branches Eliminates these stalls by guessing the target address of the branches. This work is designed to design forecast branches that operate at high operating frequency while achieving high accuracy so as to improve performance [4]. In this paper a dynamic Branch Predictor (BP) algorithm was used, as there are different types of dynamic BP exists the bimodal algorithm is used in the paper and this algorithm was applied to a soft core processor called MicroBlaze(MB).

2. Types of branch prediction strategies

There are three types of branch prediction strategies:

1) Fixed
2) Static
3) Dynamic

2-1 In fixed branch predictor this strategy isIt is easy to implement and I assume that the branch is either irreplaceable or always taken. The advantage of a strategy that is not taken is that the wizard can continue to get instructions in sequence to fill the pipeline. This involves a minimum penalty in case of error prediction. If, on the other hand, the method that is followed is always used, the wizard will pre-fetch instructions at the target branch address. This technique was used in Intel 80486 [5].

2-2 Static branch predictors statically predict a branch to be taken or non-taken. Initially, this Strategy was motivated on the observation that most branches are taken. And have set of rules about how a branch should be predicted and these rules do not change throughout the execution of a program. As such, every time a branch instruction is seen it will always be predicted in the same way. Static branch predictors require differing amounts of information to make their prediction depending on the specific prediction technique being employed. Generally this means that the branch can only be predicted once the instruction has passed through the decode stage of the pipeline. This often results in several cycles of pipeline stalls, but less than would be seen if the branch was not predicted until the execute stage was complete, this type of branch predictor was used in Intel pentium4 [6,7].

2-3 dynamic branch prediction units take advantage of the information that becomes available at run time in order to make their expectations.This means that a branch that is seen many times during the execution of a program can be predicted one way at the start of the execution and in a different way later in the execution. Data collected at run time is used to correlate the behavior of the branch with the behavior of other branches. This in turn allows patterns in branch behaviors to be identified at run time and exploited to produce increased accuracy. Furthermore, this allows branches to be predicted with reduced information, allowing branches to be predicted at the start of the pipeline and (in the case of accurate predictions) removing all branch stall cycles [8]. Dynamic branch predictors typically achieve branch prediction rates in the range of 80% to 95% [5]. Dynamic branch predictors may require a significant amount of chip area to implement [9], especially when more complex algorithms are used. Dynamic strategy seek to run-time history in order to make predictions more accurate The basic idea is to take the previous n branch of the type of branch concerned and use this information to predict the next branch. Suggest that considering the last two executions in the section will give us more than 82% accuracy prediction for most blends [10]. After that, we got only marginal improvement. This is better than from the others view. the basic idea is to take the past n branch executions of the branch type in question and use this information to predict the next one, used two bits counter more using accurate than one bit. This techniques can be used two-state finite state machine as shown in Fig.1, taken (T), not taken (NT). The forecaster will last change his predictions from T ◊ NT or NT ◊ T very quickly, although the branch may be taken mostly or often not taken [6]. Use two bits to record the forecast history of a branch instead of a single bit, and can contain two instances of T or NT instead of one case each.

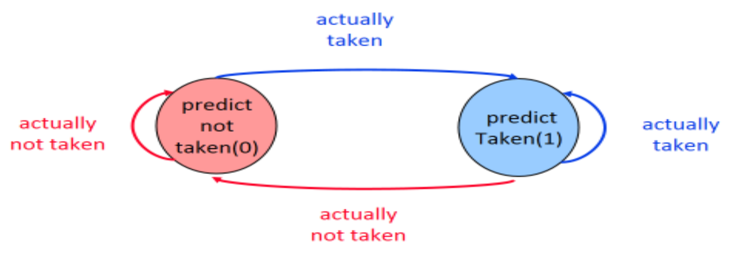

Figure 1: State machine for (1bit) counter

For tracking branch directions, 2-bit counters provide better prediction accuracies than 1-bit counters due to the additional hysteresis. For example, Represents the case 00 that predicted that the branch should be taken (left zero bits) and the branch is not actually taken (zero right), as long as the branch is not taken, remain in the case 00. If our prediction is wrong, move to case 01. However, it still predicts the "branch is not taken" as it was a mistake only once. If prediction is correct, the case is 00. If prediction is wrong again, changing prediction to “branch taken” and move to state10. This can be achieved by the fact that it always requires two consecutive false predictions of change. Adding a third bit only improves performance by a small increment. In many branch predictor designs, this incremental improvement is not worth the 50% increase in area of adding an additional bit to every 2-bit counter [11,12]. Fig 2 shows the state machine for 2-bit counter.

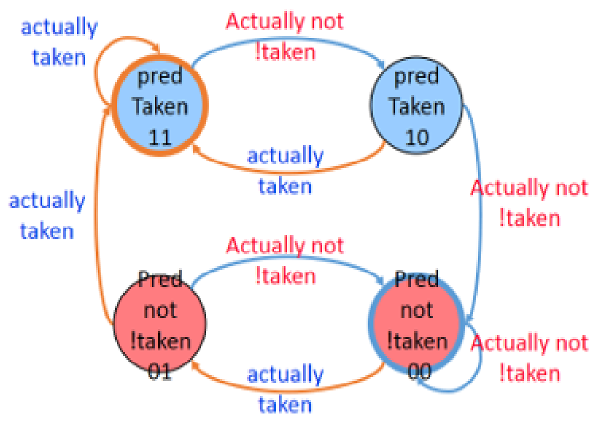

Figure 2: state machine for (2) bit counter

3. Related work

In this paper, it is clear from previous studies in dynamic branch prediction this was the center of great study in survey. One-level scheme is the center of recent studies of branch predictor algorithmYeh and Patt [13]. In the pattern history table (PHT) for two bit counters, it is indexed by a program counter of the branch address and the public record or the date of the branch [14,15]. The most significant bit counter is taken as the prediction. When we know the result of a branch, it is taken if the counter is increased if the branch is taken and reduced if the branch is not taken. The most important problem in one level prediction is aliasing [16,17] and many predictions of the newly proposed sections seek to reduce the problem of aliasing [18,19,20] but basic prediction mechanism must be not change. J. E. Smith [21] several device schemas are presented to predict the trends of the branches including the bimodal scheme. Lee and A. J. Smith has evaluated several branch prediction plans. Additionally, they have demonstrated how branch target buffers can be used to minimize delays in the pipeline that are typically encountered when branches are taken. McFarling [22] Comparison of different hardware and software methods to reduce the cost of the branch including the use of different information. Ball and Larus [23] many techniques are described to guess the trends of the most common branches at the time of translation using static information.

4. Practical Work

The branch prediction implemented with a MIPS 32 bits soft processor, specifically with a MicroBlaze processor, it is part from the embedded development kit (EDK). This MicroBlaze processor is available in Xilinx, in which it is a pipeline soft core, with two options of the pipeline depth, either 3 or 5 pipeline stages. In this research the 5 pipeline stages was used. The MicroBlaze processor is one of the core components that are used to reduce the Computer Assist Group (RISC) for executing programmable program gate arrays in the Xilinx field (FPGAs), the MicroBlaze with optional branch target prediction. A taken branch in MicroBlaze takes three clock cycles to execute. To reduce this latency (delay) micro blaze supports branches with delay slots. Used MicroBlaze to run based system on a Digilent Nexys-4 prototyping board, requires this system specific hardware architecture and the running software on system, combined these two parts in one FPGA.

Configure the Artix-7 FPGA on the Nexys-4DDR board. the hardware architecture include a processor: Micro blaze, dual ported of local memory one ports Data, and one for instructions and peripherals (IPs): UART, GPIO, timer (for performance data) and bus connecting peripheral devices with processor: AXI, clock generator and reset generator associated with the whole system,then Run Connection Automation, size of local memory is 8KB and choose for debug module (Debug & UART), and clock connection (100 MHZ),clocking wizard choose primitive MMCM with source single ended clock capable pin ,output clocks tab reset must be active low , add IP axi uartilte and we run connection automation again and check the validate design we must be successful validation and there are no errors or critical warnings in the design, the hardware architecture is done, then create the software by exporting the design in to SDK[24] .used Local Memory to 64KB.and used Cache Configuration to its default value of None. Fig. 3 shows a functional block design of the MicroBlaze core. The parts with shaded can be configurable and allowing select a specific set of features required by design but others cannot. The processor's fixed feature set includes: 32-bit general purpose registers, 32-bit address bus, single-issue pipeline, 32-bit instruction word with three processor modes and processing.

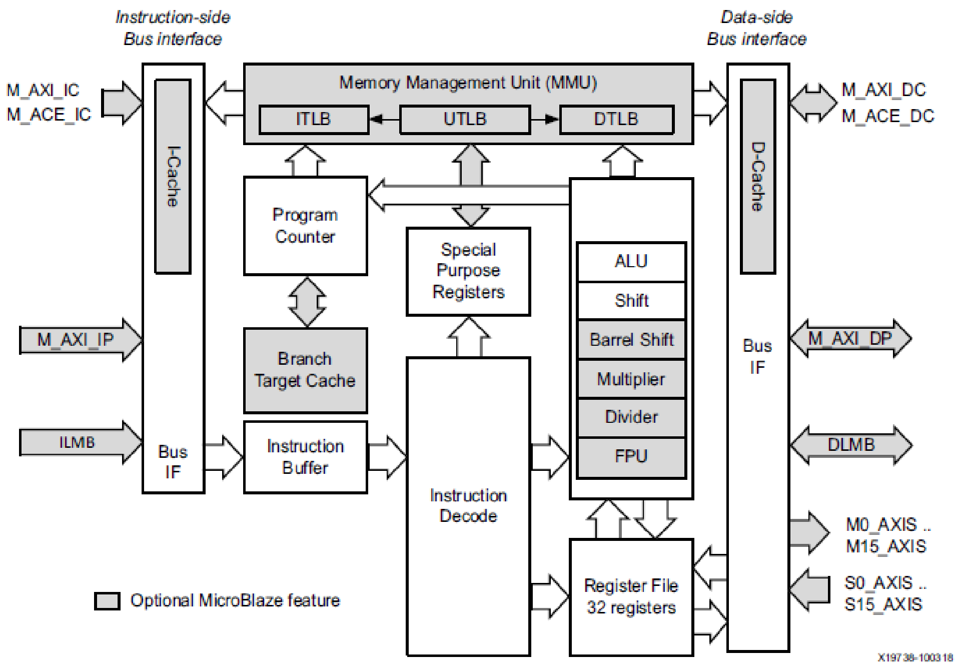

Figure 3: Block Design of Micro Blaze core

Vivado output is that part of the FPGA configuration hardware of system that describes. To emulate the hardware architecture that described in Vivado use board resources and FPGA require. Fig. 4 explain the Overview of the design and how merge the hardware means design processor and algorithm code bimodal and software (SDK) program written under (c or c++) in one configuration and then putting in FBGA board [25].

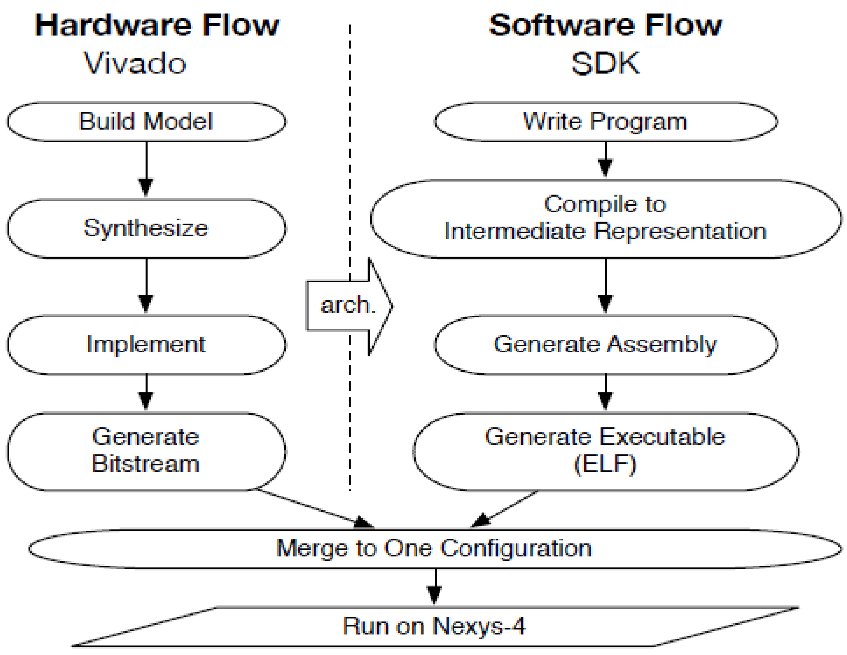

Figure 4: Overview of the Design Flow chart.

4. Global Branch Prediction

A global-history two-level predictor uses the latest branch results. These results are stored in the branch date record (BHR). BHR is a shift log where the result of each branch is converted to one end, and the oldest results are moved from the other end and ignored. The results of the sections are represented by zeros and results, which correspond to the results that are not taken and taken, so the h-bit branch record records the latest results of section h a history of the most recent branch outcomes. These outcomes are stored in the branch history register (BHR). The BHR is a shift register where the outcome of each branch is shifted into one end, and the oldest outcome is shifted out of the other end and discarded. The branch outcomes are represented by zeros and ones, which correspond to not-taken and taken, therefore, an h-bit branch history register records the h most recent branch outcomes. The branch history is the first level of the global-history two-level predictor. The second level of world-class prediction in global history is a table full of 2-bit counters. This table is called the Style History table (PHT). PHT is indexed by a series of branch address with BHR contents. The counter in the PHT indexed entry provides the prediction of the branch in the same manner as Smith's prediction (prediction is determined by the most important part of the counter). The counter updates are the same as in the Smith predictors: increase saturation in the branch taken, and subsidence in the branch is not taken. This prediction uses the results of the guidelines for the four most recent and 2-bit branches of the branch address to create an index in PHT 64 entry. With h bits from the branch date and m bits from the branch address, PHT contains 2h + m entries, when using only m bits of the branch address (where m is less than the total width of the computer), the branch address must be mapped to bits, Predict Smith. The size of the second-level expectation in the global registry depends on the total available hardware budget. For a 4K byte budget, the PHT will have 16384 entries (4,096 bytes 4 times the two-byte counters per bytes) .BRR used to initialize 11 bits four 1111, these bits are shifted left after updating all counters [3,5].

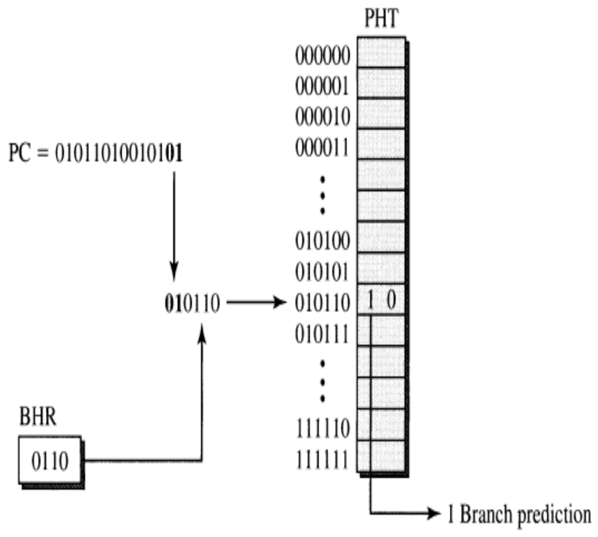

Figure 5: block design of global history two level predictor with a 4 bit branch history register

5. Results and Analysis

This paper studied the accuracy of branch prediction, at first determine the results (taken or not) for branch and if taken must know the Branch target address. The branch addresses select address depending on m (number of bits from the branch address) from the pattern History table. Inside the pattern history table explains number of entry each entry size is two bit counter. When the most significant bit is 1 the branch is predicted taken, when it is 0 the branch is predicted not taken. Every time the branch is taken the counter is incremented. Every time the branch is not taken the counter is decremented. After executing the test programs by the 5-stage pipelined MicroBlaze processor, results that shown in fig.6 and fig.7 have been gotten.

These results indicate the correctness of program execution which in turn reflects the correctness of the processor hardware design. Aliasing occurs when two different branches access the same 2-bit saturating counter in the PHT. Aliasing occurs when different branches with different outcomes try to access the same saturating counter.

This results in the counter being both incremented and decremented, such that in the worst case the two conflicting branches never receive a correct prediction which they would receive if they were using two separate counters. For this purpose the size of predictor BHT must be increased in order to avoid this aliasing problem. Global branch predictor which used the branch address and 2- bits counter the actual results. In these predictors, the predictor accuracy is function of size of BHT and achievable hit rate is 93% for program1 and 95% for program2. The following two figures explain the BHT with different size and branch history register and taking into consideration the time of development (execution) whenever the time execution is reduce the performance was the best as the fig.7and take low power consumption and reduces aliasing, Table1 summarize the different value for predictor size and execution time.

Table 1: Time execution for global history branch predictor with two program pipeline and without Pipeline with different size of BHT

Predictor size	**Time Execution For Program1 (sec)**	**Performance% For Program1**	**Time Execution For Program2 (sec)**	**Performance% For Program2**
Without predictor	**4.110**			
8	**2.102**	**66%**	**2.100**	**66%**
16	**1.500**	**73%**	**1.600**	**73%**
32	**1.226**	**77%**	**1.258**	**76%**
64	**0.964**	**81%**	**0.900**	**82%**
512	**0.632**	**86%**	**0.501**	**89%**
1024	**0.300**	**93%**	**0.209**	**95%**

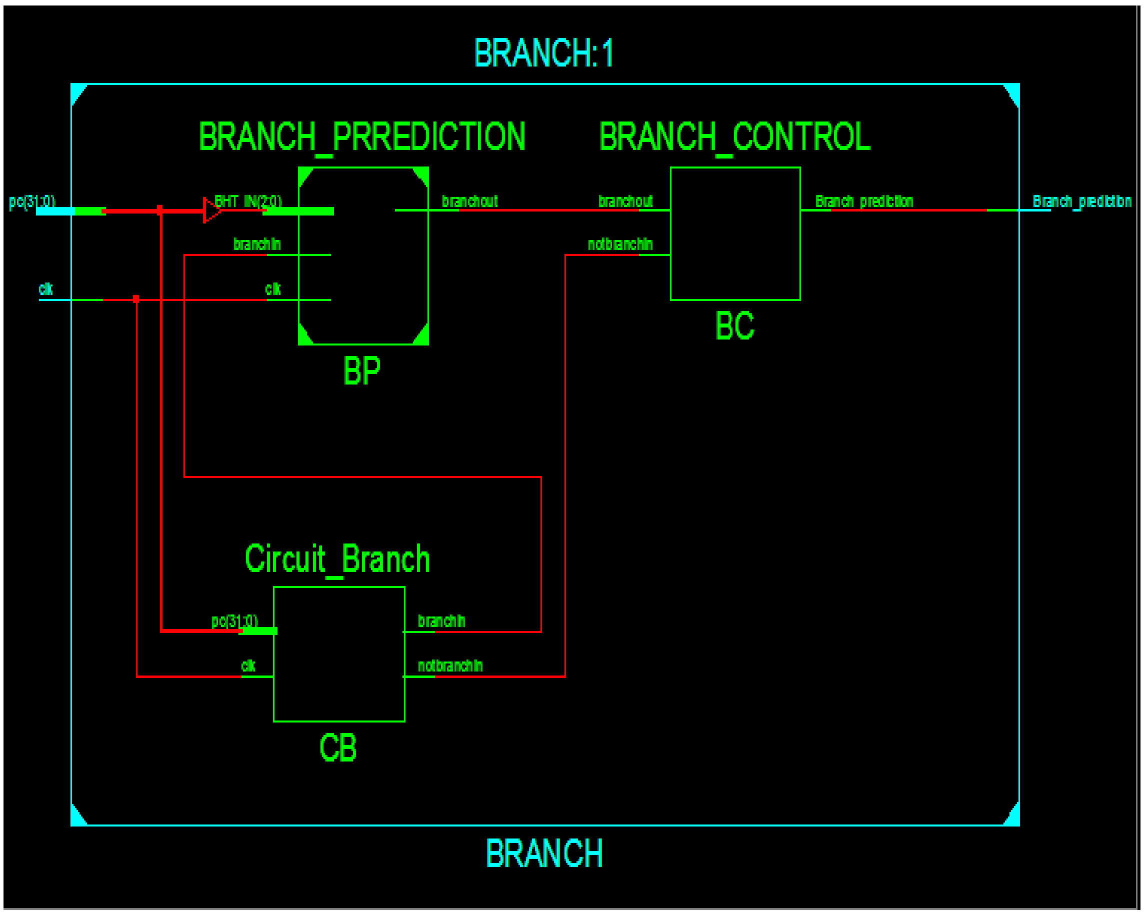

Figure 6: RTL Schematic of block design of global history two level predictor with a circuit decoder (CB) to interpret the instruction if it is a branch or no

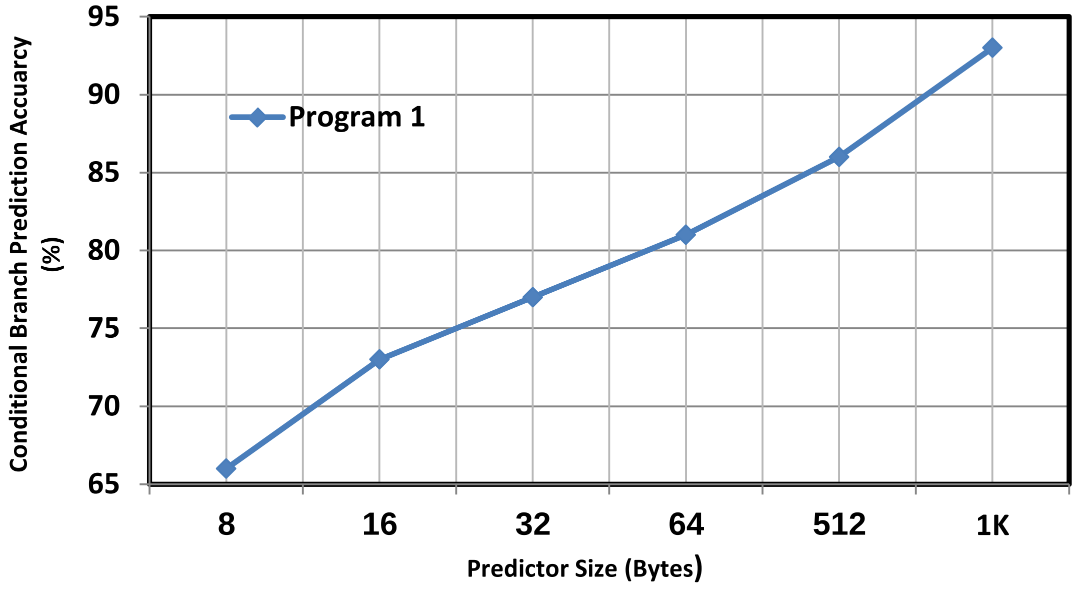

Figure 7: Global Predictor Performance for

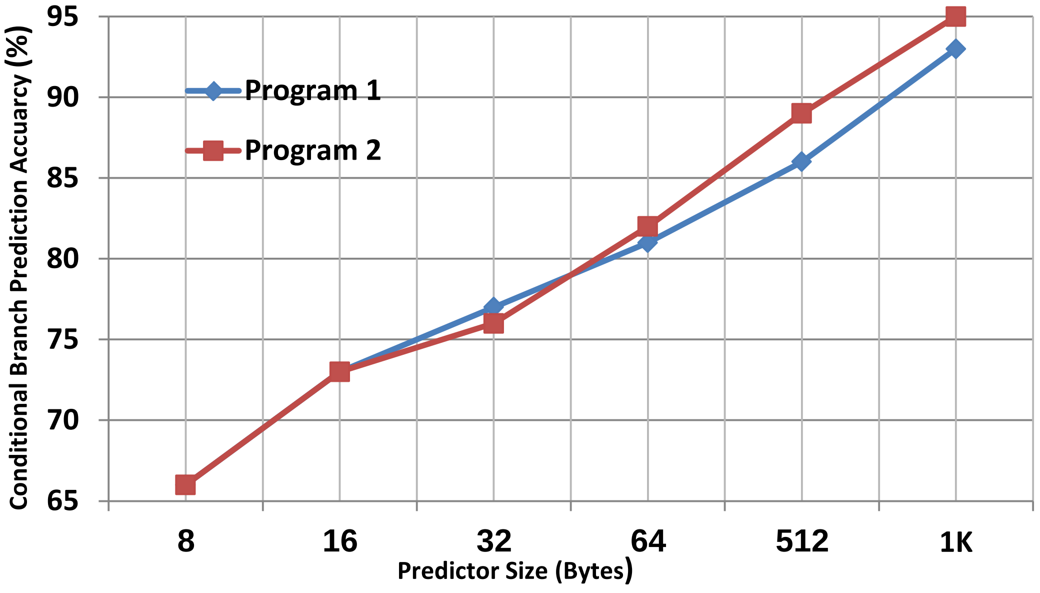

Figure 8: Global Predictor Performance for P1and P2

compares the performance of the global prediction with two program and take the best results from the curve for program2 because it achieve high performance 95% than the program1. The fig 9 shows that the global Branch predictor is more accurate thanCompared with the first initial method the bimodal branch predictor for a fixed size predictor.

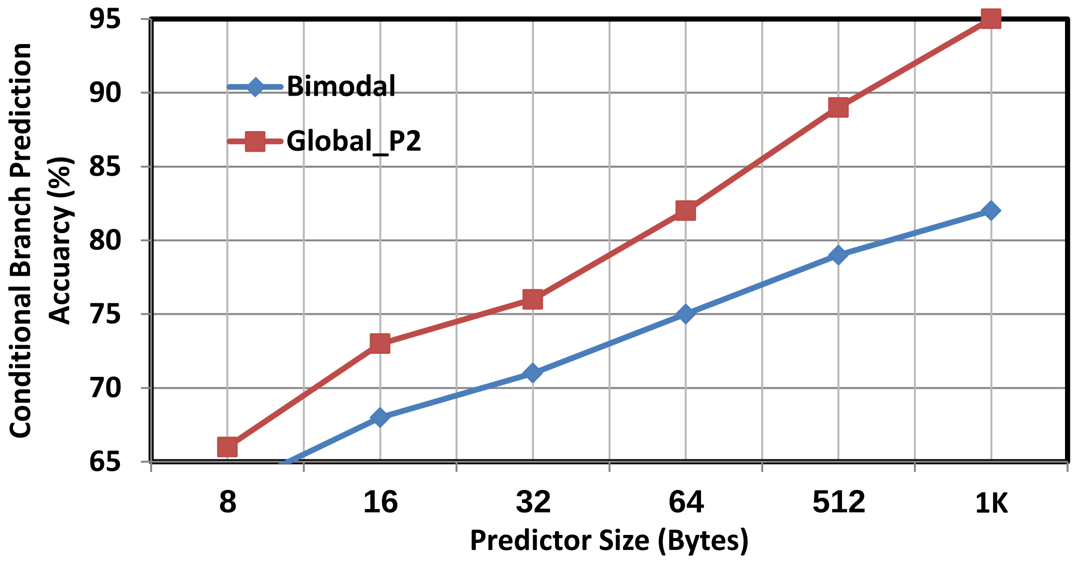

Figure 9: Global and Bimodal predictor performance

6. Conclusion

In this paper, a 5-stage, 32-bit, pipelined MIPS MicroBlaze processor has been designed in VHDL and synthesized using (Xilinx Vivado design suite 2018.1) program. This design consists of five pipeline stages which are: Fetch stage, Decode stage, Execute stage, Memory stage and Write back stage, configured on *Xilinx Vivado NEXYS 4 DDR* FPGA starter kit. It solves all the problems of branches with stalls using global dynamic branch prediction. The implementation of branch predictors for general purpose soft processors MicroBlaze it targeted to high accuracy branch predictors for pipelined processors, This paper studies the FPGA implementations of global branch predictor on NEXYS 4 DDR, and get on the best results when the predictor size reachable the max size because it prevents the repeating the same entry address (aliasing), global predictor achievable hit rate is 95%. Using more branch address bits reduces the destructive aliasing but increases the weakly biased group. The benefits of using branch addresses and PHT cannot be preserved in current two-level schemes simultaneously, but they can in the bimodal scheme.

References

[1] J. L. Hennessy and D. A. Patterson, Computer Architecture: A Quantitative Approach, 5th ed., San Francisco, USA: Morgan Kaufmann, 2012.

[2] S. P. Dandamudi, Fundamental of computer organization and design, Computer Science, 1th ed., New York, Springer, September 22, 2002.

[3] Di Wu, K. Aasaraai, and A. Moshovos, “Low-cost, high-performance branch predictors for soft processors”, in2013 23rd International Conference on Field Programmable Logic and Applications (FPL), Porto, Portugal, Sept 2013.

[4] L. Chen, S. Dropsho, and D. H. Albonesi, “Dynamic data dependence tracking and its application to

branch prediction", in High-Performance Computer Architecture, 2003. IEEE, Anaheim, CA, USA, 2003.

[5] R.Thomas , M.Franklin, C.Wilkerson, J.Stark, " Improving branch prediction by dynamic data flow-based identification of correlated branches from a large global history ", in Proceedings of the 30th annual international symposium on Computer architecture, vol. 31, no. 2, pp. 257-266, May 2003.

[6] C. Egan, "Dynamic Branch Prediction in High Performance Super Scalar Processors", Ph. D. thesis, University of Hertfordshire, August 2000.

[7] S. Sechrest, C.-C. Lee, and T.N. Mudge. "Correlation and aliasing in dynamic branch predictors". In Proceedings of the 23rd International Symposium on Computer Architecture, Vol. 24, no. 2, pp.22-32, May 1996.

[8] E. Sprangle, R.S. Chappell, M. Alsup, and Y. N. Patt. "The Agree predictor: A mechanism for reducing negative branch history interference", on conference Proceedings the 24th Annual International Symposium on Computer Architecture , Denver, Colorado, USA, June 1997.

[9] D. L. Perry, VHDL: Programming by Example, 4th ed., America: McGraw-Hill, 2002.

[10] C.-C. Lee, C.C. Chen, and T.N. Mudge. "The bi-mode branch predictor". On conference Proceedings of the 30th Annual International Symposium on Microarchitecture, Research Triangle Park, NC, USA, November 1997.

[11] Juan, L.A., et al., "Confidence Estimation for Branch Prediction Reversal", in Proceedings of the 8th International Conference on High Performance Computing. Spain, 2001.

[12] M.Abd-El-Barr and Hesham El-Rewini, "Fundamentals of computer organization and architecture". New Jersey, USA: John Wiley& Sons, 2005.

[13] S. S. Omran and I. A. Amory, "Design and Implementation of Resizable Cache Memory using FPGA". The second international Engineering Conference IEC, Ishik University, Irbil, Iraq, 2016.

[14] K. P. Singh, S. Parmar, "Vhdl Implementation of a MIPS-32 Pipeline Processor", International Journal of Applied Engineering Research, vol. 7, No.11, pp. 1952-1956, 2012.

[15] J. L. Hennessy and D. A. Patterson, "Computer Organization and Design: The Hardware/Software Interface", 4th Waltham, USA: Morgan Kaufmann, 2012.

[16] T.Olga, Steve, Chris and Rachel, "Ultra Low Power Cooperative Branch Prediction", Ph. D .Philosophy, Institute of Computing Systems Architecture, University of Edinburgh, 2015.

[17] J.W. Kwak, C.S Jhon, "High-performance embedded branch predictor by combining branch direction

history and global branch history," in Proc. IET Computers & Digital Techniques, vol.2, no.2, pp.142-154, March 2008.

[18] H. Arora, S. Kotecha, R. Samyal, "Dynamic Branch Prediction Modeller for RISC Architecture," in Proc. International Conference Machine Intelligence and Research Advancement (ICMIRA), pp.397-401, Dec. 21-23,2013.

[19] J. P. Shen, M. H. Lipasti, Modern processor design fundamentals of superscalar processors, USA, Waveland Press, 2005.

[20] N. Panwar, M. Kaur, G. Singh, "Performance Analysis of Branch Prediction Unit for Pipelined Processors", International Journal of Computer Applications, Vol. 128, no.16,pp.975-8887, October 2015.

[21] J. E. Smith. "A study of branch prediction strategies ", In Proceedings of the 8th Annual International Symposium on Computer Architecture,vol.21,no.1, pp. 135-148, May 1981.

[22] S. McFarling, "Combining Branch Predictors", TN-36, Digital Western Research Laboratory, June1993.

[23] T. Ball, and R.L. James, " Branch prediction for free", In Proceedings of the ACM SIGPLAN 1993 conference on Programming language design and implementation, New York, USA June 1993.

[24] Xilinx Inc. Micro Blaze Processor Reference Guide, July 2012.

[25] Xilinx Inc. 7 Series FPGAs Overview, Feb.