An Energy Efficient TLB Design Methodology

Dongrui Fan, Zhimin Tang, Hailin Huang
Institute of Computing Technology
Chinese Academy of Sciences
P.O.Box 2704-25, Beijing, China 100080
Telephone: +86-10-62565533-9315
{fandr, tang, huanghl }@ict.ac.cn

GuangR. Gao
Dept. of Electrical & Computer Engineering
University of Delaware
140 Evans Hall, Newark, Delaware 19716
Telephone: +1-302-8313241
ggao@capsl.udel.edu

ABSTRACT
This paper researches Translation Look-aside Buffer (TLB) of embedded processor. Based on an analysis of design-related factors: power, area, critical path and performance of our research model—Godson-I, a low-power TLB design is proposed without sacrifice of performance and timing. Using this method, the following results are achieved: power of TLB-RAM reduces 92.7% and area of TLB-RAM reduces 50%. Compared with other methods, the hit rate of this design is much higher and the accessing conflict to RAM between ITLB and DTLB is much reduced. Although our work targets to Godson-I, the proposed methodology should be applicable to other designs.

Categories and Subject Descriptors
B.3.2 [**Memory Structures**]: Design Styles – *Associative memories.* B.7.1 [**Integrated Circuits**]: Types and Design Styles – *VLSI (very large scale integration).*

General Terms
Design, Experimentation, Performance.

Keywords
TLB, low-power consumption, Godson-I, single-port RAM, energy efficient, embedded processor design.

1. INTRODUCTION
Designers of embedded architecture are paying increasingly more attention to the power and area of chip design, because they are also the deciding factors for the mobility and price of an embedded product. That is critical to survive in the industry full of competition.

This paper proposes a power and area reduction scheme in the design of the translation look-aside buffer (TLB). We illustrate our method using Godson-I processor – an embedded processor designed at ICT [2,3,12]. We improve the TLB design through three steps. Our method can reduce power and area, while keeping the new design from sacrificing of its performance and timing. We have performed various experiments and analysis to study the effectiveness of the proposed TLB design method.

Using the new TLB design method, the area of RAM part of TLB reduces 50%, and the power of RAM reduces 92.7%; the total area of TLB reduces 23.7% and the total power of TLB reduces 28.5%. This has not cause any notable timing increase and performance degradation. The experimental results show that the proposed method is both practical and effective.

ISLPED'05, August 8-10, 2005, San Diego, California, USA.

2. RELATED WORK AND EXPERIMENTAL METHODOLOGY
A TLB plays a major role in MMU (Memory Management Unit) providing the caching function to speed up the translation between virtual addresses to physical addresses. The power consumption and area of TLB construct important part of a processor. A TLB is often designed as a full-associative or set-associative buffer, which is constructed with a content-indexed CAM and an address-indexed RAM. There are different ways to structure TLBs in processor architecture design. For example, instruction TLB (ITLB) and data TLB (DTLB) can be combined into a unified TLB, or else they can be designed as separate TLBs [1,8]. In recent years, many new design methods are introduced to reduce the power consumption of TLB. The work described in [7] turns off some inactive entries of TLB according to history; [5] and [11] introduce micro-TLB, which divides TLB into two levels, and controls the number of entries accessed each time based on power considerations. But the hit rate of this method is much lower. Dividing TLB into banks [6] and Semantic-Aware Multilateral Partitioning [4] are also effective to reduce power consumption of a TLB. But these new methods are more complex, and may not be suitable for an embedded architecture design.

In this paper we report our research on an experimental TLB design in the context of the Godson-I processor. We modify the Godson-I architecture to incorporate our new TLB design and implement the design at the register transaction level (RTL). We are able to conduct an evaluation of our design at the gate level using the commercial EDA tools for power, area and timing analysis. These tools can accurately capture the effect of our new design. Several aspects of the new design are considered in our integrated experimental environment. After simulation and verification, we collected experimental data in terms of performance, power, area and timing. Suitable embedded TLB design is raised according to tradeoff and analysis on these data.

Figure 1 illustrates the original TLB design of Godson-I. When TLB miss happens, refill operation will occur to both ITLB and DTLB to keep the consistency between them. Advantage of this design is to support simultaneously accessing between instruction fetching and data visiting. Also this design is simple to realize, and easy to keep higher hit rate. But the shortcoming of the original design is that it may incur higher power and area. For ASIC design, IP cores and standard cells are widely used. SRAM is a standard IP (RF1SH from Artisan in Godson-I) and is easy to get from foundries, but CAM is constructed using standard cells in Godson-I. So CAM and RAM are separately designed in different manners.

The experimental environment is described as below. The experimental platform is Sun Blade 2000 server and SunOS Solaris 5.8. We use NCVerilog (version: v03.30.(p001)) from Cadence company to conduct simulation and use Design Compiler (version: design_vision 2003.12) from Synopsys company to do synthesis and timing analysis. VCS (vcs script version: 6.0, compiler version: 6.0.1) and Power Compiler (version: 2003.12)[9,10] from Synopsys are used to estimate power consumption. The EDA tools mentioned above are popular commercial products. The results from these EDA tools are always much more accurate than micro-architecture simulators available to us.

In our experiments, typical benchmarks, like LinuxKernal, Dhrystone, Whetstone, and Paranoia etc. are used to test the influence to performance and power for different TLB designs.

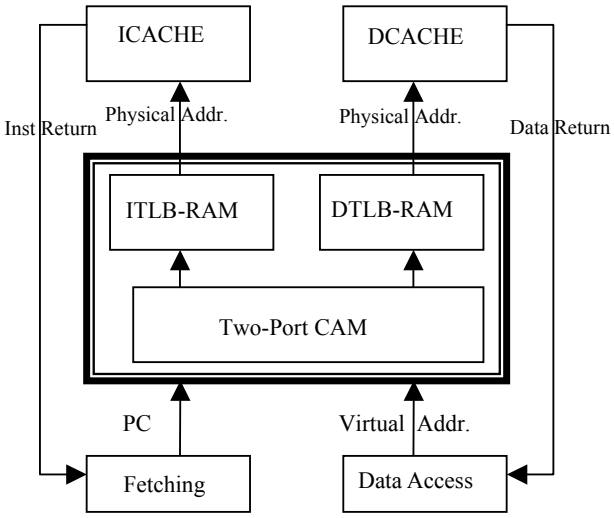

Figure 1. Original TLB design method of Godson-I

3. CHALLENGES IN EXISTING TLB DESIGN

In this section, we first analyze power, area, critical path, and performance of existing TLB design. Through the quantitative analysis we can identify the problems and possible solutions.

3.1 Power Analysis

From Figure 1, we can see that frequently accessing to CAM and RAM may significantly increase the power consumption of TLB.

Figure 2 illustrates the power distribution of Godson-I. TLB occupies 12.9% of total power, and becomes the most important source of power consumption besides CACHE. Table 1 shows the accessing frequency of TLB. ITLB ratio is the number of cycles of accessing ITLB divided by the total running cycles for each benchmark. And DTLB ratio is number of cycles of accessing DTLB divided by the total running cycles.

From Figure 2 and Table 1, we can find the power consumption of TLB is mainly due to frequent access to CAM and RAM of TLB. For instance, 34.2% accessing ratio for ITLB and 9.0% accessing ratio for DTLB are very high for these benchmarks, which are not intensive in memory accesses. On the other hand, the area of TLB is aggressive because CAM and RAM have special circuit structures. These factors all illustrate why TLB consumes so much power.

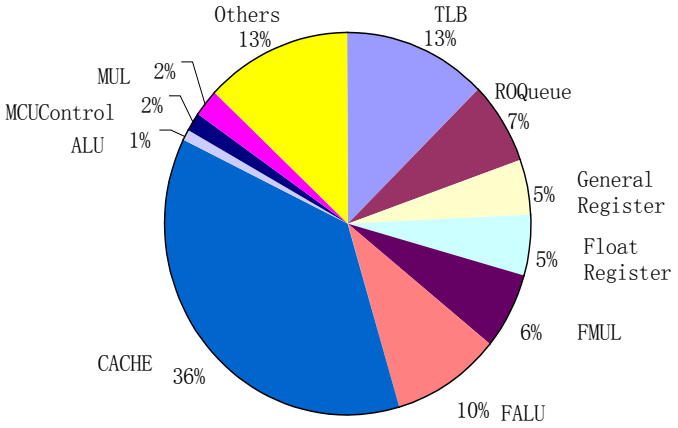

Figure 2. Power Distribution of Each Part of Godson-I.

3.2 Area Analysis

As is shown in Figure 3, area of 48-entry TLB equals to the whole area of Godson-I processor except CACHE and float point units. The graph "A" of Figure 3 illustrates area distribution of each unit of Godson-I. Area of TLB occupies 13% of the total processor. The graph "B" shows the ratio of area of each part of TLB itself. RAM constructs 50% area of TLB. If some method can reduce RAM effectively without notable impact on other factors, e.g. performance. That will be a good way to reduce the area of TLB.

3.3 Critical Path Analysis

In processor design, critical path always lies in MMU and CACHE, so does Godson-I. Figure 4 illustrates the critical path: from physical address, which is from the output of TLB-RAM, to tag comparison in CACHE module. The result of tag comparison indicates whether a CACHE hit happens. If CACHE hit happens, this operation will return data and new memory accessing operation will be allowed in. If CACHE miss happens, CACHE will send an accessing request to memory.

Because it takes much long time to access a big RAM, and compare and control operations also take extra time, this path actually becomes the critical path. So the new TLB design should

Table 1. Accessing Frequency of ITLB and DTLB for original TLB design

Benchmark	LinuxKernal	Whetstone	Dhrystone	Paranoia	Aver. Ratio
Numb. Of ITLB accessing	3,459,885	22,378,257	37,709,705	14,700,174	— —
Numb. Of DTLB accessing	974,210	6,832,384	8,768,139	3,796,648	
Running Cycles	15,986,000	66,257,750	72,926,250	49,673,750	
ITLB ratio	21.6%	33.8%	51.7%	29.6%	34.2%
DTLB ratio	6.1%	10.3%	12.0%	7.6%	9.0%

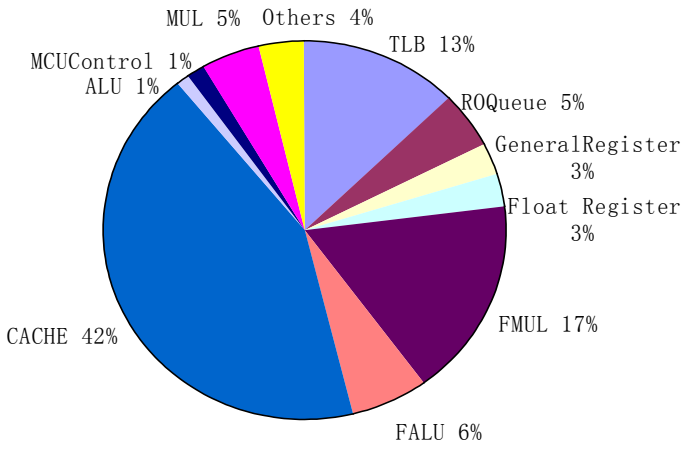

Figure 3A. Ratio of Area of Each Part of Godson-I Processor

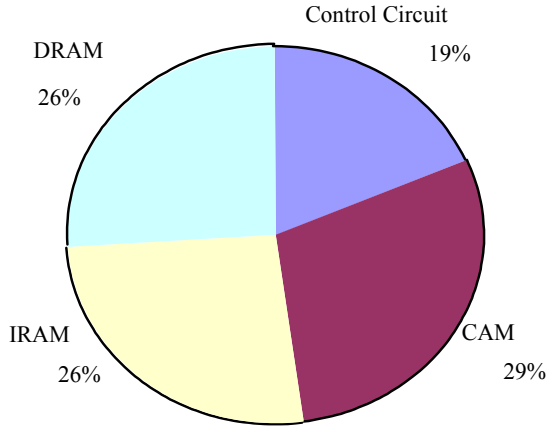

Figure 3B. Ratio of Area of Each Part of TLB

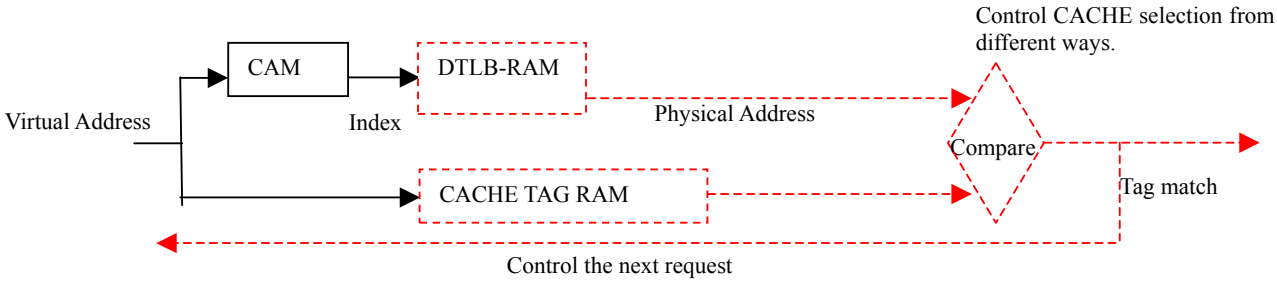

Figure 4. The Red and dashed Path is the Critical Path of original Godson-I.

not add much delay on this path. Otherwise, worse timing is unacceptable.

3.4 Performance Analysis

Because both instruction fetching and data accessing need to translate virtual address by using TLB, TLB must be able to process two visits at the same time to avoid port violation. In Godson-I processor, as shown in Figure 1, there is a two-port CAM, which can process two translation requests at the same cycle. After comparing with each entry of CAM, ITLB and DTLB generate its own index separately. Using the index, ITLB-RAM and DTLB-RAM can be queried to generate physical address separately. Thus, instruction fetching and data accessing will not conflict for port of RAM.

Although, the original TLB design keeps higher hit rate and performance, it incurs higher power and area. In the following section, we propose a new design method to resolve this problem, while keeping the performance unaffected.

4. ENHANCED TLB DESIGN

According to the above analysis, we introduce our improved TLB

Table 2. The Amount of RAM Accessing Reduces Dramatically After the First Step Improvement

Benchmark	Linux Kernal	Whetstone	Dhrystone	Paranoia
Original ITLB: Amount of RAM Accessing	3,459,885	22,378,257	37,709,705	14,700,174
Enhanced ITLB: Amount of RAM Accessing	932	227,410	693,290	33,838
ITLB: Enhanced Amount/Original Amount	2.7%	1.0%	1.8%	0.2%
Original DTLB: Amount of RAM Accessing	974,210	6,832,384	8,768,139	3,796,648
Enhanced DTLB: Amount of RAM Accessing	13,427	1,553,550	1,973,773	245,482
DTLB: Enhanced Amount/Original Amount	1.4%	22.7%	22.5%	6.5%

Table 3. Power Saving for the First Step Improvement

Benchmark	Linux Kernal	Whetstone	Dhrystone	Paranoia	Aver. Reduction
Power Reduction of RAM:	98.0%	88.2%	87.9%	96.7%	92.7%
Power Reduction of TLB:	21.3%	28.9%	36.9%	26.9%	28.5%

design method in three steps. The first step is to reduce power; the second step is to reduce area; and the third step is to reduce delay on critical path. At last, a new TLB design is structured to meet the need of embedded system.

4.1 The First Step: Reduce Power

Analysis in the previous sections shows that power consumption of TLB is serious. The aim of the first step is to reduce power. The method is described in two parts as following sentences. Firstly, when a new memory accessing comes, the virtual address of this accessing is compared with the previous virtual address of last accessing. If the two addresses lie in the same page, RAM of TLB won't be accessed and the previous physical tag of the last access is used for this memory accessing. Furthermore, this compare logic is processed simultaneously with CAM query operation to avoid extra delay on critical path. Because the last accessing information will be kept in registers inherently until a new accessing comes, no extra hardware logic is needed to realize this method. Secondly, we utilize some high bits of the virtual address to judge which address space it belongs to. If the address lies in unmapped address space, we can map the address directly by using algorithm and don't access the RAM of TLB.

We don't access the RAM of TLB until it's necessary. Thus, the dynamic power of RAM can be controlled to a lower level, and the RAM can stay in low-power state for a longer time.

Because of the data locality, continuous accessing to a specific page is very popular. We utilize this property in ITLB and DTLB. Though comparing with historical accessing is not a new concept, e.g. introduced in [7], our design method is still different on realization. We compare it simultaneously with CAM comparison to avoid extra delay on critical path. The first step constructs a solid foundation for step two.

The amounts of accessing the RAM of ITLB and DTLB are listed in Table 2 to compare the effectiveness with original design. After the first improvement, the amount of accessing ITLB is only 0.2% ~ 2.7% of the original design, and the amount of accessing DTLB is only 1.4% ~ 22.7% of the original design. Accordingly, the power of TLB will be reduced dramatically. From Table 3, we can see that the power of RAM has been reduced by 98% for LinuxKernal. And the average power reduction of RAM is 92.7%. Because the power of CAM is equal to the original design, the average power of the total TLB design is reduced by 28.5%. The first improvement is very effective.

4.2 The Second Step: Reduce Area

As shown in Table 2, the accessing frequency to RAM reduces a lot after using the first-step improvement. Will the conflict reduce if ITLB and DTLB share only one RAM with only one port after utilizing the first-step effect? If the conflict reduces a lot, the performance will not be influenced. But using only one one-port RAM can reduce much area. Of course, we can use a RAM with two ports, but its area is nearly equal to two RAMs with one port. It is also unacceptable.

As shown in Table 4, ITLB and DTLB are very frequently accessed simultaneously for original design because both of them are accessed too much times. So two one-port RAMs or one two-ports RAM are needed to avoid conflicts and to keep performance for original design. But after using the first-step enhancement, requirement of accessing RAMs reduces a lot, and as a result, requirement of accessing simultaneously the two one-port RAMs reduce a lot. For examples, conflict of Dhrystone reduces to 0.01% of original design, and conflict of Whetstone reduces to 0.03% of original design. So we can be confident to conduct the second step: using only one one-port RAM to save power and area without sacrifice of performance.

The following experimental results can prove the efficiency of the second-step improvement. Table 5 compares the running time of the second-step improvement with original design n RTL simulation environment. We find the affect to performance is no more than 0.1%. Because a little conflict between ITLB and DTLB changes the execution behavior of the same program, that even makes Paranoia run a little bit faster - 0.02%. Anyway, the influence can be ignored.

Now, we will see how much benefit we can get from the second improvement. Because ITLB and DTLB share only one one-port RAM, the area of RAM is reduced by 50%. From Figure 3B, we can figure out that the area of TLB will be reduced by 26%. The synthesis result from Design Compiler shows that under the same

Table 4. Number of Accessing to ITLB and DTLB simultaneously

Benchmark	Linux Kernal	Whetstone	Dhrystone	Paranoia
Number of Original Design	542,612	4,406,089	7,003,155	2,202,878
Number of Enhanced Design	2	1,141	982	1,265
Enhanced Design/ Original Design	0.0004%	0.02%	0.01%	0.06%

Table 5. Impact on Performance after ITLB and DTLB share one RAM

Benchmark	Linux Kernal	Whetstone	Dhrystone	Paranoia
Running time of Original Design (ns)	63,946,000	265,344,000	294,324,000	198,661,000
Running time of Enhanced Design (ns)	63,946,000	265,031,000	291,705,000	198,695,000
Performance Degradation	0.00%	0.12%	0.90%	-0.02%

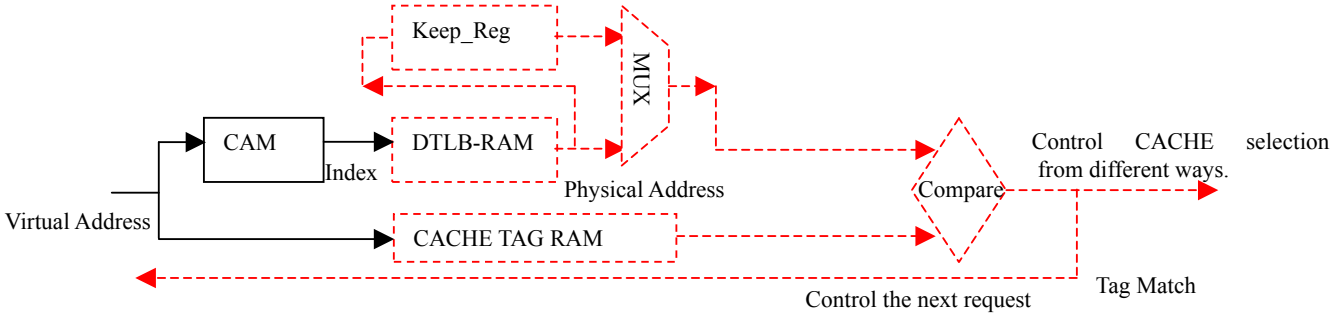

Figure 5. The structure of the circuit using the second-step improvement

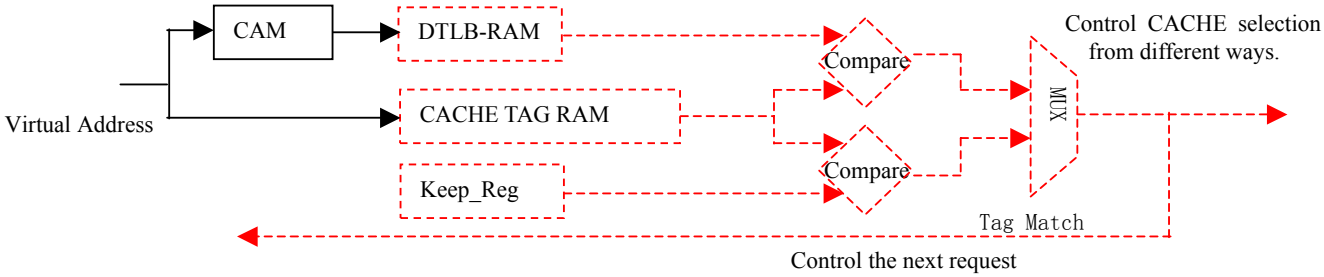

Figure 6. The structure of circuit after using the third-step improvement

constrains with 0.18um process, the area of the original TLB design is 724,464.8um^2, and it becomes 552,445.9um^2 after using the second-step improvement. The area of TLB reduces 23.7%. The effectiveness of step two is obvious. Furthermore, it is certain that power consumption will be reduced to a lower level because of less area.

What's the shortcoming of the second step? It is inevitable that the modification adds extra delay on the critical path. We will analyze one scenario to describe this problem. First, ITLB is accessed by a PC (Program Counter), and the corresponding physical tag is got from the RAM; Second, DTLB is accessed by another virtual address, the output of the RAM changes to a new physical tag; Then, ITLB is accessed by another new PC, which lies in the same page with previous PC, and the new PC will not access RAM according to the first-step improvement. Thus, we cannot use the output of RAM directly, and we must add extra registers to lock the previous physical tag. If a virtual address lies in the same page with previous virtual address, the physical tag kept in the registers will be used. If a virtual address doesn't lies in the same page with previous virtual address, RAM will be accessed and output of RAM will be used. As shown in Figure 5, a MUX is added to select from outputs of RAM and Keep_Reg. Keep_Reg is the register set to keep the previous physical tag.

According to the design specification, the output of TLB-RAM is 128-bit width. Selection on the two 128-bit data must bring heavy load and long delay. The timing results of Design Compiler show that the delay increases from 3.09ns to 3.26ns. It is unacceptable to add so much delay on critical path. How to resolve this problem? The third step is introduced in the following section to uncover this answer.

4.3 The Third Step: Reduce Delay

The area and power of TLB are reduced a lot because of the step one and step two. But these improvements bring a new trouble—extra delay on critical path. This is what this section should try to resolve.

Because the critical path lies in DTLB, and the design of DTLB is similar with ITLB, we focus on the structure of DTLB to exploit an effective way to resolve the timing problem. The reason of extra delay on the critical path is that the width to be selected is too wide and the load is also too heavy. So we specially design the related circuit as shown in Figure 6. Selection between RAM and Keep_Reg registers is postponed. Outputs of DTLB-RAM and Keep_Reg are both compared with the tag of CACHE. Then we get two one-bit results. Selection between the two results can be done to judge whether it is a CACHE hit or not. Though the extra delay is also added, it is a selection between two one-bit signals, not two 128-bit signals. The synthesis results from Design Compiler show that the timing of critical path is 3.11ns after using the third-step improvement, and only 0.02ns is added compared with original 3.09ns.

As a result, using three steps of improvements, the power and area of TLB of Godson-I are reduced to a lower level without sacrifice of timing and performance. Of course, there are much more details needed to be considered in the real design of Godson-I processor. In this paper we introduce the main method of the improvement after realizing the total design in the real chip.

5. SUMMARY AND FUTURE WORK

In this paper, we have proposed a new TLB design method and analyzed its effect on performance, power and area on the Godson-I processor architecture. We have implemented our proposed design and reported preliminary results. We have demonstrated the merit of a judiciary combination between the use of a single-port RAM and some history information. Furthermore, this single-port RAM is shared by ITLB and DTLB without influence the performance and timing, while keeping higher hit rate.

In the future, we plan to investigate a configurable TLB design method that can optimize power, area, timing and performance with different applications on embedded systems.

6. ACKNOWLEDGMENTS

This work was supported by Fundamental Research Foundation of ICT (20056020), the National High-Tech Research and Development 863 Program of China (2005AA110010), the National 863 International Cooperation (20041080).

7. REFERENCES

[1] ARM Limited. ARM1020T™ (Rev 0) Technical Reference Manual. www.arm.com, 2004.

[2] Dongrui Fan, Hongbo Yang, Guangrong Gao, Rongcai Zhao. Evaluation and Choice of Various Branch Predictors for Low-Power Embedded Processor. *Journal of Computer Science & Technology, Vol.18, No.6, pp833-838,* Nov. 2003.

[3] Dongrui Fan. Low Power Technology Research for Godson Processor—master to Ph.D. thesis. CAPE memo7, http://cape.ict.ac.cn, July 2002.

[4] Hsien-Hsin S Lee, Chinnakrishnan S Ballapuram. Energy Efficient D-TLB and Data Cache using Semantic-Aware Multilateral Partitioning. *ISLPED, pp306-311,* 2003.

[5] J. B. Chen, et al. A Simulation Based Study of TLB Performance. *Proc. 19th Symp. Comp. Arch., pp114-123,* 1992.

[6] Jung-Hoon Lee, Gi-Ho Park, Sung-Bae Park, Shin-Dug Kim. A selective filter-bank TLB system. *ISLPED pp312-317,* 2003.

[7] Lawrence T Clark, Byungwoo Choi, Michael Wilkerson. Reducing translation lookaside buffer active power. *ISLPED pp10-13,* 2003.

[8] MIPS Technologies. MIPS32® 4Kc Processor Core Data Sheet. www.mips.com, 2004.

[9] Synopsys Inc. Power Compiler Reference Manual. Version 2003.

[10] Synopsys Inc. Power Compiler User Guide. Version 2003.12.

[11] T Juan, et al. Reducing TLB Power Requirements. *ISLPED, pp196-201,* 1997.

[12] Weiwu Hu, Zhimin Tang. The Architecture of Godson-I processor. *Journal of Computer，Vol.26，No.4，* April 2003.