2002 International Computer Symposium (ICS2002)
Workshop on Computer Systems

Title: Equipping the SIMD MAC Operations into embedded RISC Processors

Authors: L. Wang, Leo Fang and H. Y. Hsu

L. Wang (***corresponding author***)
(Associate Professor in the Department of Electrical Engineering at Feng Chia Univ.)

Address: Department of Electrical Engineering,
Feng Chia Univ. Taichung, Taiwan 407, R.O.C.

Email-address: lwang@plum.iecs.fcu.edu.tw

Tel: 886-4-24517250-3838 Fax: 886-4-24516101

Leo Fang (Graphics Driver Engineer in VIA TECHNOLOGIES, INC.)

Address: 2F, No.9, Li-Hsin Rd. V, Science-Based Industrial Park, Hsin Chu, Taiwan, R.O.C.

Email-address: **g8912158@knight.fcu.edu.tw**

Tel: 886-3-6667336-338 Fax: 886-3-6667337

H. Y. Hsu (Associate Professor in the Department of Information Engineering at Feng Chia Univ.)

Address: Department of Information Engineering,
Feng Chia Univ. Taichung, Taiwan 407, R.O.C.

Email-address: **hyhsu@fcu.edu.tw**

Tel: 886-4-24517250-3756 Fax: 886-4- 24516101

Equipping the SIMD MAC Operations into embedded RISC Processors

L. Wang[+] Leo Fang [++] and H. Y. Hsu [+++]

[+] Department of Electrical Engineering in Feng Chia University

[++] VIA TECHNOLOGIES, INC.

[+++] Department of Information Engineering Feng Chia University

Email : lwang@plum.iecs.fcu.edu.tw, g8912158@knight.fcu.edu.tw, hyhsu@fcu.edu.tw

Abstract

Because of the variety and popularity of IA products, the techniques of SOC that satisfy the restrictions of small-size, low-power consumption, and cost effective design have became the main concern for microprocessor design. However, we observed that some services that are performed by massive computation like data encryption, video presentation have merged into the use of low-end IA devices. Thus the research decide to modified a simple embedded processor to satisfy the demands of the dedicated applications by equipping the processor with some important hardware features.

The paper targets the applications of multimedia for investigation. By observing the program behavior of multimedia applications, some important operations be executed frequently in these programs are introduced. According to the analysis results, the paper attempt to enhance the execution of MAC operations in a loop by the technique of SIMD to promote the performance. The parallel SIMD MAC instruction, named PMLAV, can speedup the execution of multimedia applications efficiently with a minor hardware overhead.

By simulating the execution of multimedia applications to verify the efficiency of the added instruction. We have proved that the addition is a cost effective design for embedded processors.

Keywords: Embedded processor, Information appliances, Multimedia, SIMD, Instruction set architecture

I. Introduction

The dramatic growth of IA(Information Appliances) market has changed the design trend of computer system. Because of the variety and popularity of IA products, several design philosophy for computers have been modified. For example, performance is not the most important consideration for micro-processor design, the techniques of SOC(System On Chip) that satisfy the restrictions of small-size, low-power consumption, and cost effective design have dominated the focus of micro-processor design. An embedded processor is built in the kernel of SOC, to provide the capability for processing dedicated applications efficiently under restrict hardware constraints.

The emergence of embedded processor was due to the requirement of the IA applications. The design of an embedded processor must fit the following features:

1. The chip size must small enough to fit IA products. Meanwhile, the limit of power consumption also makes the processor to be built with simpler hardware.
2. Embedded processor is designed to support some specific applications such as multi-media [1][2][3] and communication usage efficiently. On the contrary, the considerations for the applications like text editor and file manipulation are not important.
3. For providing good quality of IA services, the processor must process massive multi-media video and audio data in real time [4].
4. The life cycle of most IA products is much shorter than traditional computers. An expensive processor core is not acceptable. The fact means that the constraint of hardware cost is crucial for the design.

The following is a brief survey about the architecture features of some well-known embedded processors. We can conclude the important features for an embedded processor by examining these processors:

1) Intel Strong ARM and Xscale family [5]:

The processor is an embedded processor chip launched by Intel on 1998. It is an 32-bits processor which support ARM v4 instruction set architecture [6] that featuring:

(1) The chip is implemented by super CMOS and low-voltage process technology to achieve optimal performance/power ratio (MIPS/mW rate).

(2) Its RISC instruction set architecture can achieve highly pipeline execution rate.

(3) A high memory bandwidth is achieved by a 16K/8K bytes split instruction/data cache, and a special designed 512-bytes mini-data cache.

In the second generation, Xscale, released on 2000. Besides the improvements on cache capacity and implementation technology, it introduced superscalar and SIMD parallel technology [7] into the processor:

(1) A branch target buffer is added to provide the ability of dynamic branch prediction.

(2) A co-processor with 6 SIMD instructions to support voice/image signal processing is equipped into the chip.

(3) A simple memory management circuit is inserted to support the translation from virtual to physical address.

2) TI OMAP architecture [8]:

The processor was proposed by TI(Texas Instruments) for the 2.5G and 3G wireless applications. The chip uses a basic ARM RISC Processor to handle the traditional data calculation and program control; an extra DSP is included for multimedia data and digital signal processing. It's an embedded processor with dual kernel engines to offer optimal working efficiency and power saving ability.

3) NEC V830R architecture [9][10]:

NEC V830R is an embedded multimedia processor for processing the Real-time audio and video data. It is used for digital TV and multimedia devices. As shown in figure 1, an extra 64-bits media unit is designed for elevating the process of multimedia data. It is designed by SIMD technology with parallelism of 4 to process 16-bits multimedia data in parallel. The architectural characteristics of NEX V830R includes:

(1) Two-way superscalar with 6 stages pipeline in integer unit.

(2) Media unit can execute data with SIMD features.

(3) A split instruction/data caches both with 16K bytes are built in the processor.

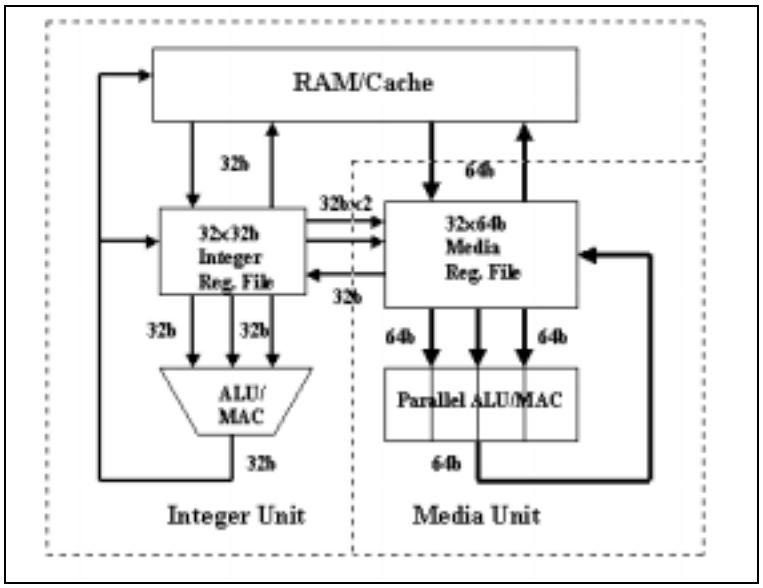

Figure 1 NEC V830R [10]

From the brief survey described above, we can classify the embedded processors into two categories: simple RISC embedded processor as a low-end product, and expensive embedded processor built by a RISC processor conjunction with a high-performance co-processor as a high-end product. The two types of processors are designed for different purpose. For example, low-end embedded processor is applied for the IA devices like cellular phone and personal PDA. The devices are equipped with simple and cheap elements to support the basic functions of communication. On the other hand, high-end embedded processor is applied for the devices that are used for professional and complicate applications. Notebook is an example that built by the high-end processor to support the functions of multi-media processing and professional CAD calculation.

From the viewpoint of applications, we observed that some services that are performed by massive computation like data encryption, video presentation, web-base computation have merged into the use of low-end IA devices. Although the high-end

embedded processors can support the execution of the applications, the cost and the size are not acceptable. In order to propose a solution that can balance the requirement of the usage and the cost of the system. We decide to develop an embedded processor based on the low-end products that can satisfy the demands of the complex applications by equipping the processor with some important hardware features.

We target the applications of multimedia as the first step for investigation. By observing the program behavior of multimedia applications, some important operations be executed frequently in these programs are introduced. These operations have been redesigned as added instructions and implemented into a RISC processor core.

ARM7TDMI [11][12] is the embedded processor selected as the base architecture for the study. The processor is a famous product that share a great part of the IA market. It is widely used as the processing core of cellular phone, personal PDA, and hand-top game player. The added instruction proposed by this study has been merged into the microarchitecture of ARM7 in gate-level design. By simulating the execution of multimedia applications to verify the efficiency of the added instruction. We have proved that the addition can improve the performance of the execution of multimedia applications without complicated hardware overhead.

This paper is organized as: Section 2 analyses the characteristics of multimedia applications and introduces the idea for enhancing the performance is then proposed. Section 3 introduces the instruction format and pipeline architecture of the added parallel SIMD [13][14][15] instruction, the consideration for hardware implementation is also included in the section. The simulation results shown in the section 4 exhibits the efficiency of this research, and a concluding remark is made in the section 5.

II. An observation for multimedia applications

For upgrading the execution performance of multimedia applications from the viewpoint of instruction set architecture, the behavior characteristics of related applications must be analyzed first.

The benchmarks used for analysis are:

1. Colorspace: It is a functional program to converse the colors of an image.
2. Composite: It is an image composition function that can generate a new image from two or more images.
3. Convolve: It is a filtering program for voice or image data.
4. Edge detection: It is a function that drawing the edges in a picture.

We analysis the 4 benchmarks and conclude the common characteristics as described below:

1. The data be processed in the multimedia program is smaller than 32-bits, most of them are 8-bits pixels and 16-bits audio data.
2. Most of instructions be executed in a multimedia program are contributed from a small kernel. For example, there are totally 2556285 instructions be executed in the Convolve benchmark to filter a 226K bytes image. 95.8% of these instructions are executed from a kernel function, nv_filt(), which is a small function with repeated loops.
3. The most frequently used operations in the applications are multiply, add, and multiply-accumulate(MAC). For example, the processing of sampling and filtering voice data in DSP use convolution function frequently. The function is a series of MAC operations as shown below:

$$y_n = \Sigma w_i x_{n+i} = \Sigma W_k X_{k,\ n}$$

As another example, the DCT(Discrete Cosine Transform)/IDCT(Inverse Discrete Cosine Transform) algorithms used in voice and graphic data compression/ decompression are both completed by multiply and repeated MAC operations:

$$y(k) = \alpha(k) \Sigma x(n) \cos(\pi(2n+1)k / 2N),\ k = 0,\ 1,\ \ldots,\ N-1$$

$$x(n) = \Sigma\ \alpha(k)\, y(k) \cos(\pi(2n+1)k / 2N),\ n = 0,\ 1,\ \ldots,\ N-1$$

$$\alpha(k) = \sqrt{1/N},\ k = 0\ ;\ \alpha(k) = \sqrt{2/N},\ k \neq 0$$

According to the analysis results, this study attempt to enhance the execution of

MAC operations in a loop by the technique of SIMD to promote the performance. By examining the ISA(Instruction Set Architecture) and microarchitecture of ARM7TDMI, we find that the parallel SIMD MAC instruction, named PMLAV, can speedup the execution of multimedia applications efficiently with a minor hardware overhead.

The operation of PMLAV is shown in Figure 2. Unlike the operation of MAC instruction, PMLAV will treat a 32-bits data as a couple of 16-bits multimedia data and execute them in parallel. Although the parallelism degree is only increased to two, the performance will be improved evidently for the applications.

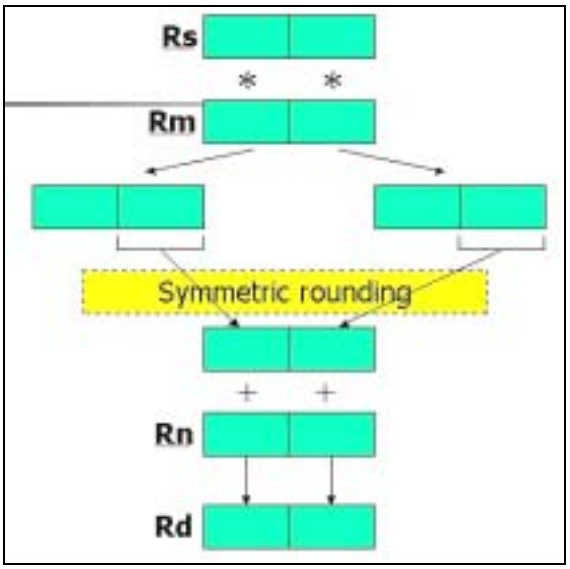

Figure 2 Operation of PMLAV

We take a simple benchmark -- Dot, as an example to show the improvement: Assuming there are two arrays: a[3]={1,2,3} and b[3]={4,5,6}, are defined for the Dot operation, the C language codes of the operation are list below:

```
#include<stdio.h>

int main(void)
{
   short a[3]={1,2,3},b[3]={4,5,6};
        int sum=0;
        int i;
        for(i=0;i<3;i++)
```

```
    {
        sum+=(a[i]*b[i]);
}
    printf("sum=%d",sum);
    return 0;
}
```

The assembly codes of the for loop created by the integrated developing environment compiler, ADSv1_1, are shown below:

Mem. Address	Object Codes	Assembly Codes	Line
0x000080d8:	e28d200c	ADD r2,r13,#0xc	..1
0x000080dc:	e0822080	ADD r2,r2,r0,LSL #1	...2
0x000080e0:	e28d3004	ADD r3,r13,#4	….3
0x000080e4:	e0833080	ADD r3,r3,r0,LSL #1	 4
0x000080e8:	e1d220f0	LDRSH r2,[r2,#0]	...5
0x000080ec:	e1d330f0	LDRSH r3,[r3,#0]	...6
0x000080f0:	e0211293	MLA r1,r3,r2,r1	.…7
0x000080f4:	e2800001	ADD r0,r0,#1	.8
0x000080f8:	e3500003	CMP r0,#3	….9
0x000080fc:	bafffff5	BLT 0x80d8	..10

The four instructions in the front of codes, line 1 to line 4, are executed for the calculation of the memory address. The following two instructions, line5 and line 6, will then load the contents of arrays into the dedicated registers for the MAC operation. When the MAC operation is completed as shown in line 7, the instructions in line 8 to line 10 are executed to control the execution flow of the loop.

As shown in the following assembly codes, by modifying the operand fields of instructions in line2 and line4 to increase the displacement of the array, and load two data

at a time by replacing the LDRSH instructions to LDR instructions in line5 and line6. The PMLAV instruction list in line 7 will execute 2 MAC operations simultaneously and lead the iterative execution of this code segment be reduced from three to two:

```
Line   Assembly Codes
 1   ADD     r2,r13,#12
 2   ADD     r2,r2,r0,LSL #2   /*Displacement changed to 4
 3   ADD     r3,r13,#4
 4   ADD     r3,r3,r0,LSL #2   /*Displacement changed to 4
 5   LDR     r2,[r2,#0]
 6   LDR     r3,[r3,#0]        /*LDRSH is replaced by LDR
 7   PMLAV   r1,r3,r2,r1       /*MLA is replaced by PMLAV
 8   ADD     r0,r0,#1
 9   CMP     r0, #2            /*Number of loop executed changed to 2
10   BLT     0x80d8
```

The idea of SIMD has widely adopted in the design of high-end processors. Intel's MMX technology, NEC V830R multimedia processor, Intel's Xscale processor are examples that use the technique of SIMD to improve the processing of multimedia data. It is noted that although the idea of SIMD proposed in this paper is the same as the processors list above, the design goal is quite different. A complete SIMD circumstance to execute operations in parallel is not adequate to the design of a low-end embedded processor because of the hardware constraints. This study focus on the selection of the most frequently executed operations from the multimedia applications, and equips the SIMD feature to support these operations for a RISC processor with minor hardware modification. In the following section, we will introduce the details of the hardware manipulation for the PMLAV instruction.

III. The consideration for hardware implementation

In order to add the feature of parallel MAC operation, PMLAV, into the core of ARM7 processor, the architecture of ARM7 is examined at first. The datapath of ARM7 is shown in Figure 3, besides the circuit for thumb, it can be divided into seven parts:

1. Register Bank: There are thirty-two 32-bits data registers and six status registers in the bank. All registers have two read ports and one write port to access data except R15. R15 is the program counter with three read ports and two write ports.

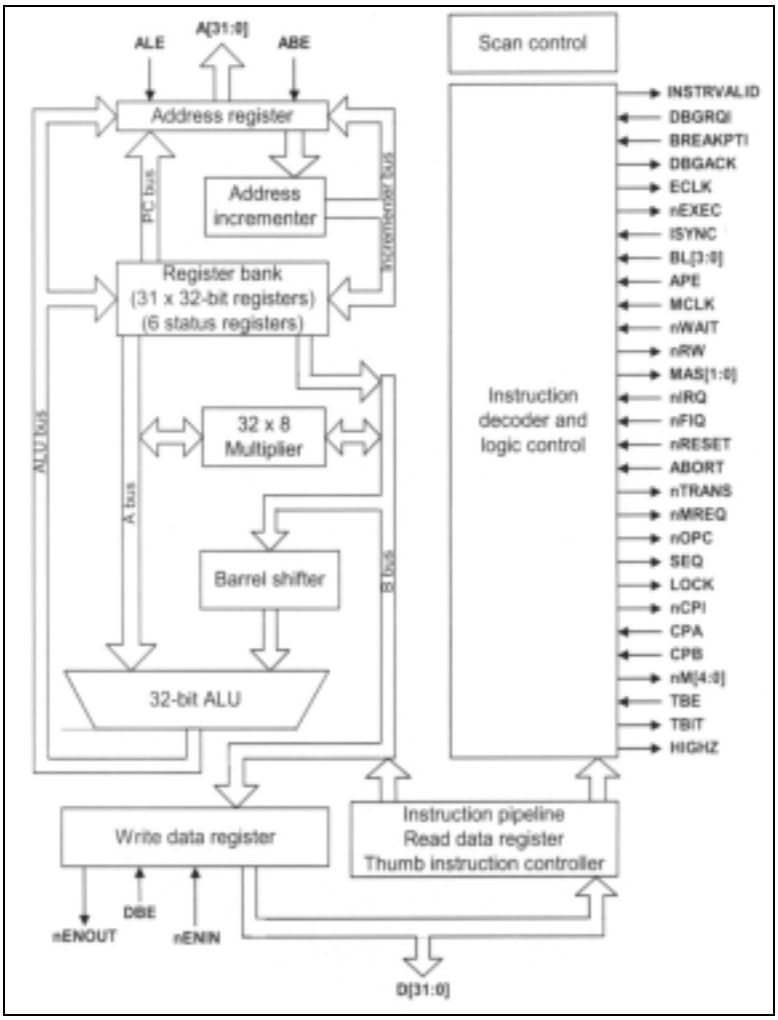

Figure 3 The microarchitecture of ARM7[11]

2. Multiplier: The processor can multiply or multiply-accumulate 32-bits integer data by means of Booth's algorithm. The multiplier can realize 32*8 multiplication every cycle and achieve the 32-bits multiplication in four cycles.

3. Barrel Shifter

4. ALU: The processor performs arithmetic and logic calculations in the ALU unit.
5. Address Register and Incrementer: The address register and incrementer are used to latch the memory address and produce the content of the next program counter.
6. Write/Read Data Registers: The write/read data registers are two latch interfaces for store and load operations.
7. Instruction Decoder: The control unit of the processor.

The pipeline of ARM7 is arranged to be three stages as list below:

1. Fetch Stage: The instruction be dedicated by R15 is fetched into the instruction decoder.
2. Decode Stage: The executed instruction is decoded to create the control signals for the following execution cycles.
3. Execute Stage: It is the most complex stage with multiple cycles. The operations of this stage are proceed according to the sequence of register reading, shifting, ALU calculating, to register write.

In this paper, we add the new PMLAV instruction to enhance the execution of multimedia applications. The instruction format of PMLAV is designed based on the format reserved in ARM's ISA, The operation codes 010S and 011S in bit 20 to bit 23 of the ARM instruction format are two reserved instructions that can be used for the PMLAV instruction as exhibited in figure 4:

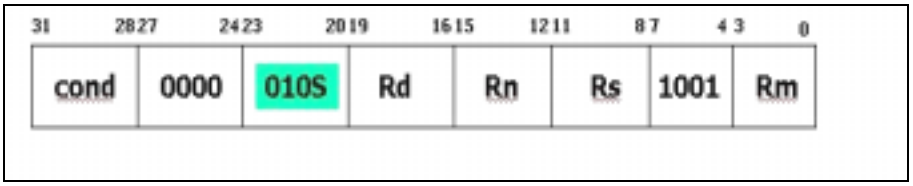

Figure 4 PMLAV instruction format

In original pipeline arrangement for MLA instruction, there are 6 cycles are required to complete the operations of MLA as shown in figure 5. It means that if we can arrange the execution of PMLAV into the 6 cycles, there will be no extra latency created from the view of pipeline. As a matter of fact, due to the characteristic of Booth algorithm used in

the multiplier, the cycles spent for the execution of PMLAV is not increased but reduced to 4 cycles without lengthening the cycle time.

Figure 5 MLA pipeline and RAW latency

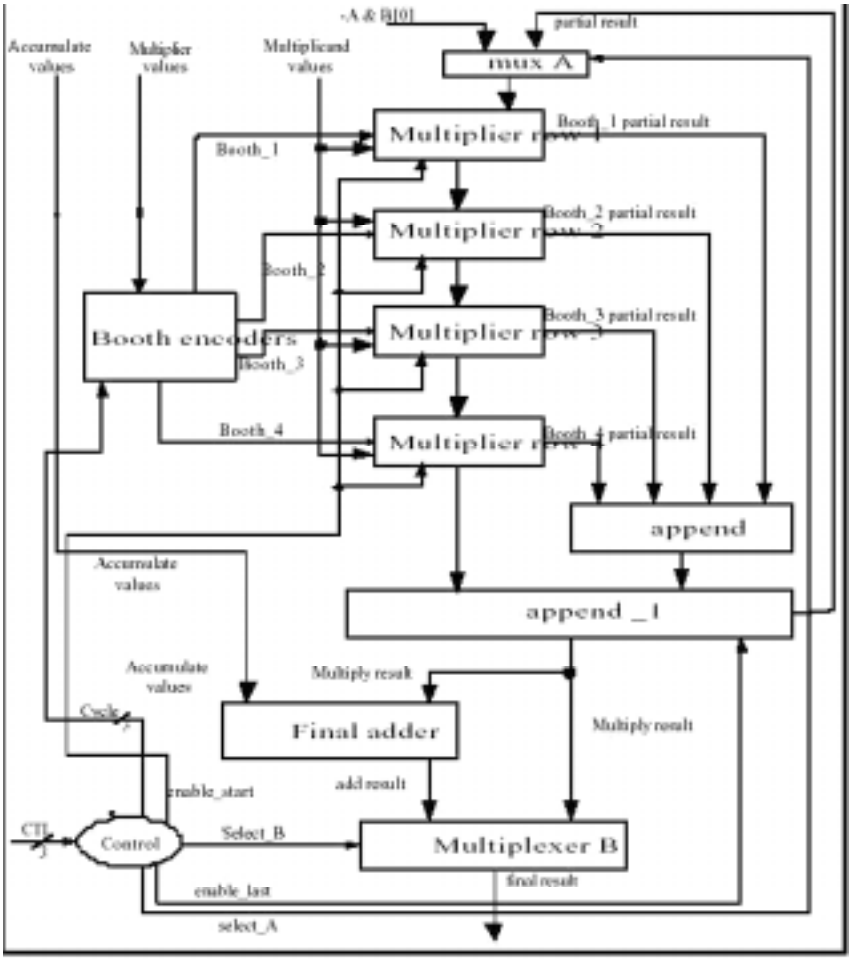

Figure 6 32*32 Multiplier [16]

The main difference between the PMLAV and MLA instructions is on the function of multiply that PMLAV multiply two pairs of 16-bits data in parallel and MLA proceed the multiplication as one pair of 32-bits data. The multiplier proposed by I. J. Huang and Y. L. Hung [16] is designed to speed up the execution of ARM7's multiplication by using

a improved Booth algorithm. It is used as the basis for the design of PMLAV. Figure 6 is the block diagram of the multiplier.

The multiplier is organized with four 8-bits multiplier, there are multiplier row-1 to row4, and conducts four cycles to complete 32×32 bits multiplication. The details of this multiplier are examined by the research and some modifications are made for PMLAV instruction. Because of the limitation of the paper length, only two parts that play the crucial roles for the design are introduced in this paper. Figure 7 and figure 9 show the internal circuits of Booth encoder and Append_1. The elements circled by dotted line in the figures are added hardware for achieving SIMD function. For controlling the execution, the control unit is requested to send a SD control signal to select the function of original multiplication or SIMD parallel multiplication.

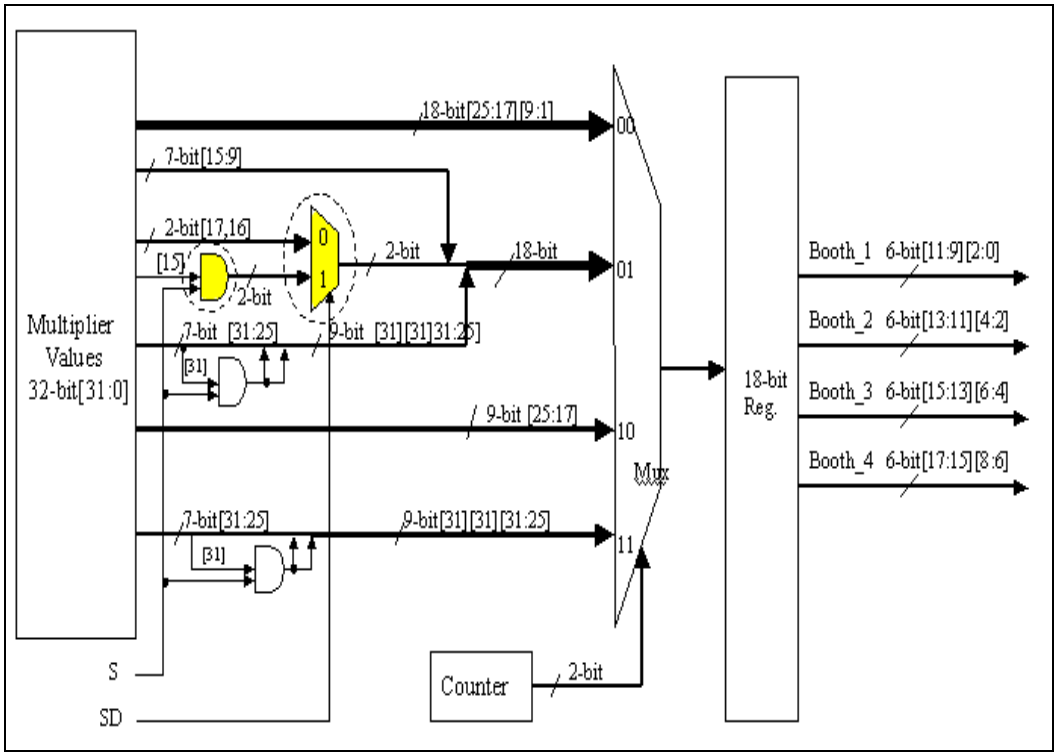

Figure 7 Circuit designed for Booth encoder

1. Booth encoder

The Booth encoder separates the 32-bits multiplicand into four 8-bits data, send different 6-bits of them in each cycle to the circuits of multiplier row-1 to row-4 in parallel. It will direct the four multipliers to perform the correct function according to

the definition list in figure 8. By adding an AND gate to perform the sign extension for the 16-bits data and a two-to-one multiplexer to select the bits for original MLA/multiply instructions and the bits for PMLAV instruction in the second cycle, the modification is completed.

Multiplier			Operation	Remarks
B[i+1]	B[i]	B[i-1]		
0	0	0	0	String of zeros
0	0	1	+A	End of 1's
0	1	0	+A	A single 1
0	1	1	2A	End of a 1's
1	0	0	-2A	Start of 1's
1	0	1	-A	End/start of 1's
1	1	0	-A	Start of 1's
1	1	1	0	String of 1's

Figure 8 The directors for the multiplier-row

2. Append_1

As shown in figure 9, the circuits in the left part of the figure will count the timing and send the enable signals to the four latches to latch the corresponding bits from the Booth partial result created by the four multipliers in each cycle. As mentioned above, while executing the PMLAV instruction, the operations can be completed within two cycles. During the first cycle, latch1 and latch3 will be enabled simultaneously. Similarly, latch2 and latch4 will be enabled simultaneously in the second cycle. So we add two OR gates to help the enabling of latches. The two added two 2-to-1 multiplexers, MUX B and MUX D, are inserted to select the 8 bits Booth partial result, and send them to latch3 and latch4. Furthermore, an OR gate is added to help the selection for whether the output of the 66-bits data are the final result or the temporal data that must be fed back.

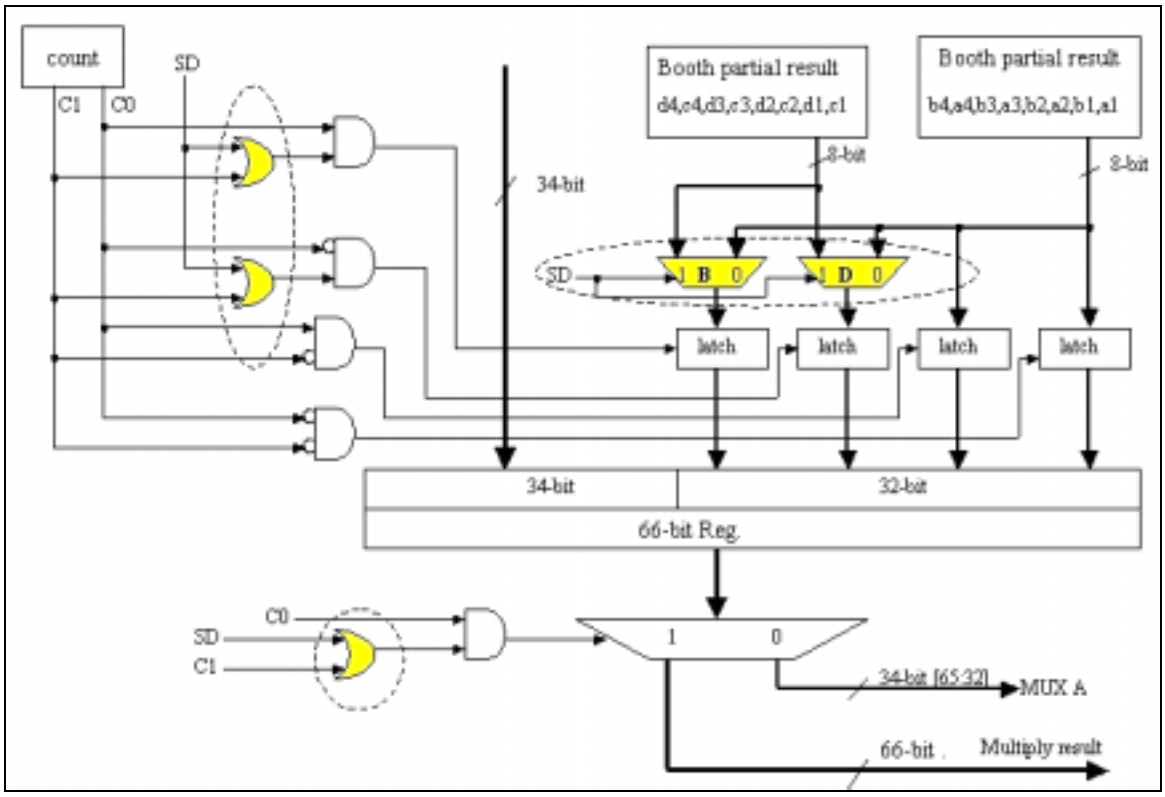

Figure 9 Circuit designed for Append_1

The whole modifications for the multiplier are completed by adding fourteen 2-to-1 multiplexers, one 4-to-1 multiplexer, one 8-to-1 multiplexer, four AND gates, three OR gates, and five NOT gates. It shows that the feature of added PMLAV instruction can be equipped into the ARM processor by adding a few logical gates. In the next section, the performance improvement gained by the added instruction is evaluated via simulation.

IV. Performance Evaluation

This research has develop a ARM7 simulator for performance evaluation. As shown in figure 10, by fetching the assembly codes produced by the ARM developer suit, the simulator can simulate the execution of ARM7 and report the simulation results such as the number of cycles for program's execution, the number of times for each instructions, the usage rates for all registers, etc.. It is noted that because of the modification is done by hand, only the kernel loops of benchmarks are modified for simulation. By replacing the MAC instructions in the loops with the proposed PMLAV instructions, and modifying the related load and compare instructions to maintain the

semantics. The codes are then simulated by the same simulator to statistic the related data for comparison.

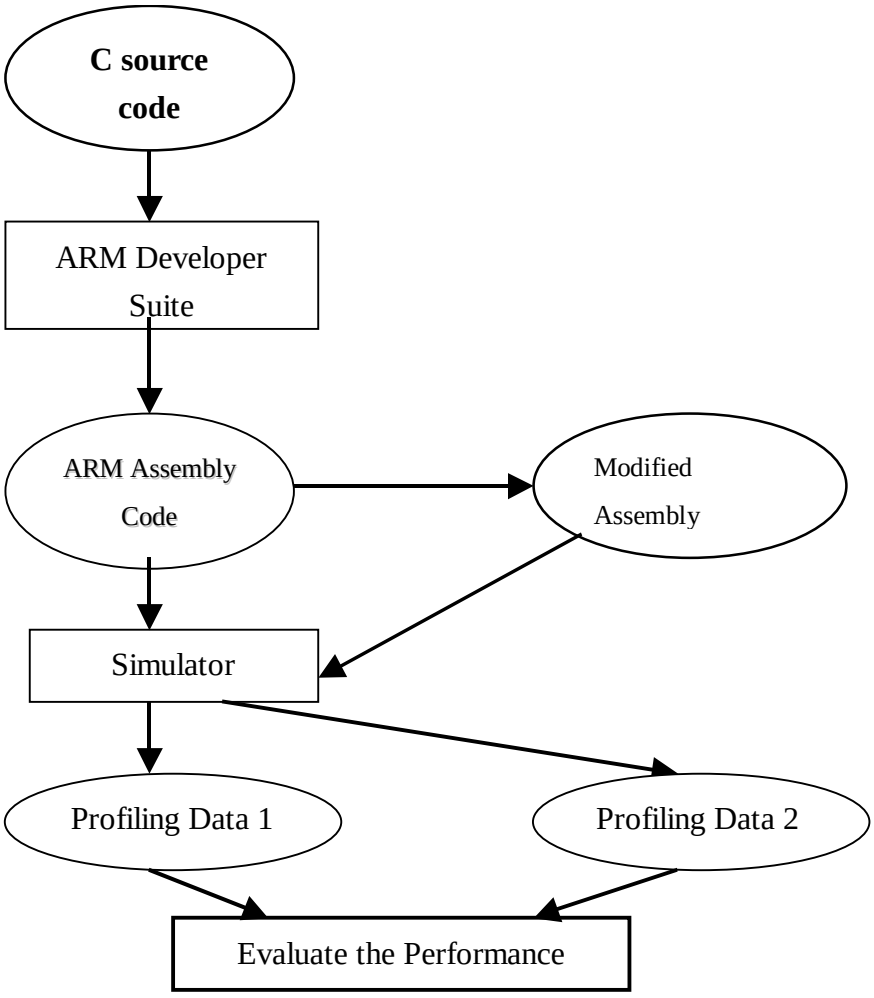

Figure 14 Compiling and simulation procedure of ARM

The function of Dot described in section II is first simulated for evaluation. Because of the execution of Dot function is dominated by the for loop with MAC operation inside, we can expect that the performance improvement can reach to 50%, it is an ideal situation for the benchmark. Table 1 is the number of instructions be executed and the number of execution cycles for the two versions of Dot function. It shows that the instructions be executed can be reduced by 33% and lead a speedup to 1.48. It is consistent with the expectation of the study.

Table 1 Simulation results produced by Dot function

Dot function	Original version	Enhanced version
Total instructions executed	31	21
Total cycles for execution	71	48

The four multimedia applications, Colorspace, Composite, Convolve, and Edge detection, introduced above are then simulated, the simulation results are exhibited in Table 2.

Table 2 Simulation results produced by the four benchmarks

Colorspace	Original program executed without the enhancement	Kernel executed without the enhancement	Kernel executed with the PMLAV enhancement
Total instructions executed	1178260	833735	717012
Total cycles for execution	2757827	2084556	1816093
Convolve	Original program executed without the enhancement	Kernel executed without the enhancement	Kernel executed with the PMLAV enhancement
Total instructions executed	2556285	816795	686109
Total cycles for execution	4858823	1552840	1278399
Composite	Original program executed without the enhancement	Kernel executed without the enhancement	Kernel executed with the PMLAV enhancement
Total instructions executed	956281	822402	666146

Total cycles for execution	2409782	2072453	1731815
Edge detection	Original program executed without the enhancement	Kernel executed without the enhancement	Kernel executed with the PMLAV enhancement
Total instructions executed	3254512	867870	749840
Total cycles for execution	7810828	2084700	1789625

From the table, we can find that the execution of the kernel loops share an evident ratio of the total execution time. The execution ratio of the kernel loops ranges from 27% to 86%. By replacing the MAC operations with the parallel PMLAV instructions to enhance the execution of the kernel. The performances for the kernels are improved from 15% to 21% and lead the execution of the whole benchmarks to gain the improvements from 11% to 20%.

V. Concluding Remarks

The study notice that some applications that are performed by massive computation like data encryption, video presentation are going to be merged into the use of low-end IA devices such as cellular phone and personal PDA. Although the high-end embedded processors can support the execution of the applications, the cost and the size are not acceptable. The paper dedicates that by carefully embedding some important features of high-end or ILP [17] or SIMD processors into a RISC embedded processor, the performance of the applications can be improved without inducing a heavy hardware overhead.

We target the applications of multimedia for investigation. By observing the program behavior of multimedia applications, the MAC operation is filtered for enhancing. The new instruction, PMLAV, is proposed that can multiply and accumulate two multimedia data in parallel. The paper shows that by adding a few logical gates into

the processor core to provide the feature of PMLAV instruction, the instruction will improve the performance of multimedia applications by 10% to 20%. It is noted that the operation of multiply instruction could be equipped with the same parallel feature without extra overhead since the modification of multiplier is done. However, the efficiency of the addition needs to be further investigated.

The idea is also being used to enhance the efficiency of the data encryption algorithms. In the other study held by the same research, a special table lookup instruction is proposed that can improve the performances of the encryption algorithms from 12% to 34%. The instruction can be applied by modifying the function of shifter in ARM7 with a few hardware overhead, too. Both investigation results prove that the performance of an embedded processor can be improved by equipping the special designed instructions from the view of ISA.

References

[1] Wong, S.; Cotofana, S.; Vassiliadis, S., "*Multimedia enhanced general-purpose processors*", 2000 IEEE International Conference on Multimedia and Expo, ICME 2000, Volume: 3, Page(s): 1493-1496.

[2] Diefendorff, K.; Dubey, P.K., "*How multimedia workloads will change processor design*", Computer, Volume: 30 Issue: 9, Sept. 1997, Page(s): 43-45.

[3] Conte, T.M.; Dubey, P.K.; Jennings, M.D.; Lee R.B.; Peleg, A.; Rathnam, S.; Schlansker, M.; Song, P.; Wolfe, A., "*Challenges to combining general-purpose and multimedia processors*", Computer, Volume: 30 Issue: 12, Dec. 1997, Page(s): 33-37.

[4] Lee, R.B., "*Multimedia extensions for general-purpose processors*", IEEE Workshop on SIPS 97 - Design and Implementation., 1997, Page(s): 9 –23.

[5] Intel Corporation, "*The Intel® Microarchitecture and Its Role in the Intel® Internet Exchange™ Architecture*", http://developer.intel.com/design/iio/papers/273429.htm.

[6] David Seal, "*Architecture Reference Manual*", 2nd Edition, Addison Wesley Longman Inc, 2000.

[7] D. A. Patterson and J. L. Hennessy, "*Computer Architecture: A Quantitative Approach*", 2nd Edition, San Mateo, California: Morgan Kaufmann, 1996.

[8] "*OMAP™ Technology Overviews*", Document Number: SWPY001, TI White Paper, Issued: Dec. 2000 Copyright Texas Instruments Ltd.(TI), 2000.

[9] Suzuki, K.; Arai, T.; Nadehara, K.; Kuroda, I., "*V830R/AV: embedded multimedia superscalar RISC processor*", IEEE Micro, Volume: 18 Issue: 2, March-April 1998, Page(s): 36-47.

[10] Kuroda, I.; Murata, E.; Madehara, K.; Suzuki, K.; Arai, T.; Okamura, A., "*A 16-bit parallel MAC architecture for a multimedia RISC processor*", IEEE Workshop on Signal Processing Systems, 1998, Page(s): 103-112.

[11] Steve Fuber, "*ARM System Architecture*", Addison Wesley Longman Inc, 1996.

[12] "*ARM7TDMI Technical Reference Manual*", ARM Limited 2001.

[13] Holmann, E.; Yoshida, T.; Yamada, A.; Mohri, A., "*A media processor for multimedia signal processing applications*", IEEE Workshop on Signal Processing Systems, SIPS 97 - Design and Implementation., 1997, Page(s): 86 –96.

[14] Chia-Lin Yang; Sano, B.; Lebeck, A.R., "*Exploiting parallelism in geometry processing with general purpose processors and floating-point SIMD instructions*", IEEE Transactions on Computers, Volume: 49 Issue: 9, Sept. 2000, Page(s): 934 – 946.

[15] Gustin, V.; Bulic, P., "*Extracting SIMD parallelism from 'for' loops*", International Conference on Parallel Processing Workshops, 2001, Page(s): 23 –28.

[16] I. J. Huang; Y. L. Hung, " *Cost-Effective Microarchitecture Optimization of the ARMTDMI Microprocessor* " , Proceedings of the International Computer Symposium, Taiwan, December 2000.

[17] B. R. Rau; J. A. Fisher, "*Instruction-Level Parallel Processing: History, Overview, and Perspective*", Journal pf Supercomputing, Feb. 1993, Vol. 7, No. 1/2, Jan. Page(s): 9-50.