A Research on an ASIP Processing Element Architecture Suitable for FPGA Implementation

Li Zhang, Shuangfei Li , Zan Yin and Wenyuan Zhao
School of Electronics Engineering
Xidian University
Xi'an,China
zhang1_li2@mail.xidian.edu.cn , sfli@mail.xidian.edu.cn, zanyin@mail.xidian.edu.cn, zhaoweny100@mail.xidian.edu.cn

Abstract—**Application Specific Instruction Set Processors (ASIPs) combine the high processing speed of ASICs and the program abilities of DSPs, therefore can realize high parallelism and high processing speed with their parallel process-unit array architecture. The key problems in ASIP system designing are: the design of the programmable processing element (PE) and inter-PE communication mechanisms. A design of ASIP processing element which is suitable for FPGA implementation with RISC like architecture is introduced in the paper. The primary goal of this design is for digital image processing, but it is also suitable for the applications such as radar and communication signal processing. The carefully designed circuit structure and specific instruction set of proposed processing element make it universal, and thus can be tailored easily for many particular applications. The design of the ASIP's functional units and the circuit utilization within the FPGA are presented with great detail and the correctness of the design is verified by some application results.**

Keywords-Computer Architecture; RISC; Application Specific Instruction Set Processor; FPGA

I. INTRODUCTION

Application Specific Instruction Set Processor (ASIP) technology is a new digital signal processing system design approach developed from ASIC and DSP technology [6]. The key feature of ASIP technology is the development of a specific instruction set and architecture for a particular application or for a set of applications [6, 5]. The digital signal processing systems using ASIP approach are very suitable for them to be implemented in FPGAs, due to FPGA's reconfigurable ability, large scale circuit density and abundant in-chip functional resources such as RAMs, arithmetic cells, and clock management cells, etc.

Especially, it's rather easy to implement multiple ASIP processing elements (PE) into a single FPGA chip, thus high parallelism and high processing speed can be realized with parallel process-unit array architecture.

The programmable processing element is the core unit of the whole ASIP processing system and thus its design is one of the key problems to be solved.

This paper introduces a design of an ASIP processing element architecture suitable for FPGA implementation. The primary goal of the design is for digital image processing, but carefully designed circuit structure and specific instruction set of proposed processing element make it universal, and thus can be tailored easily for any particular applications.

II. THE ARCHITECTURE OF ASIP PROCESSING ELEMENT

As mentioned above, ASIP processing element is the core part of the whole processing system, all the relevant signal processing algorithm can be realized in the PE by the way of software programming. The main design aim of the PE circuit is: utilizes the inter FPGA's resources abundantly to realize the relevant DSP algorithm with a processing speed as high as possible. Meanwhile, the scale of the circuit should be kept as small as possible. To reach these targets, the following design problems must be solved:

- The architecture selection of the PE.
- The selection of the specific instruction set.
- The memory structure and inter - PE communication mechanism.

A. The Design of the ASIP PE Architecture

Essentially, ASIP processing element is a tiny DSP designed for a set of applications. For it to be realized in the FPGA efficiently, very carefully selection of its architecture and specific instruction set from applied algorithm is important. Current research of ASIP architecture mainly focuses on the development of Very Long Instruction Word (VLIW) structure [5, 9]. For the VLIW structure, there are many functional units (FU) in one PE and all the FUs can operate simultaneously. The inter-PE parallelism can be obtained through the development of instruction level parallelism (ILP), so a single PE can achieve very high computational density. But the circuit complexity of VLIW structure is very high, and in order to use it efficiently, specific software assembler or compiler which can explore the ILP must be developed. It is a very difficult task that needs extensive development effort. An alternative structure is the RISC architecture. This architecture has advantages in its simple system structure, high circuit resource utility, and low design complexity. Most of all, the shot design time, make it very suitable for FPGA realization.

The primary application of the ASIP processing element presented in this paper is the digital image processing. However, it is also targeted for the applications such as radar and communication signal processing and thus the selection of the architecture and relevant instruction set must have some degree of generality, and can be tailored easily for certain particular applications. A 16-bit RISC architecture shown in Fig.1 is adopted in our design. In order to lower the design complexity, a single clock cycle instruction structure is used in the design.

The proposed ASIP processing element consists of six main functional units: arithmetic unit (ALU & Mult), general purpose register set (GPRs), data address generator (DAG), program sequencer, data storage element, and data exchange register set. From circuit design point of view, there are two main signal streams in the structure: program control stream and data stream. The program control stream is designed around the program sequencer, and the decoder of each instruction is distributed in the related functional unit. The data stream is exchanged through the GPRs under control of DAG. This design style is in accordance with the design rules of RISC processor.

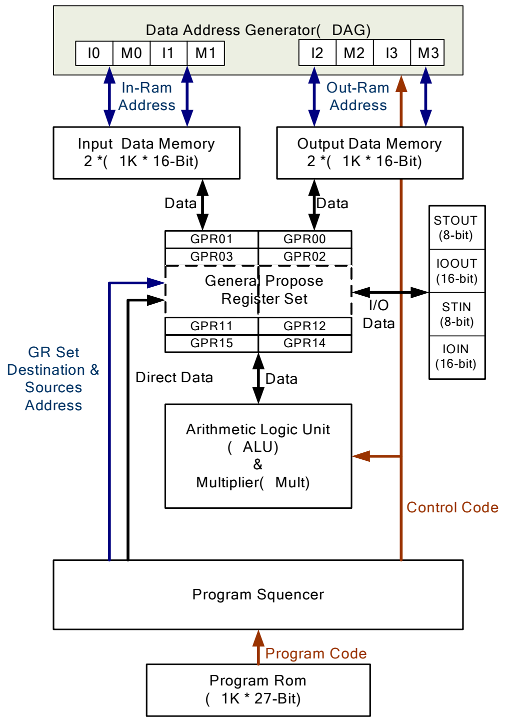
1) Arithematic Unit (ALU & Mult)

The arithmetic unit is actually an ALU with multiplier or multiply assistant circuit. All the operands and operation results except 32-bit multiply result are 16-bit. There are two approaches to construct this arithmetic unit in the FPGA:

- Adopt the configurable macro cell to build the main circuits such as adder and multiplier. The advantages of this method are: easier circuit design, very high calculation speed, and single clock cycle multiply operation. So, it can be applied in the high signal rate, and real-time processing applications. But it requires the target FPGA to have the corresponding macro cell resources, which only a few new types of FPGA manufactured with advanced technology may have. The price of these types of FPGAs is very expensive.

- Design the whole necessary circuits and implement them with FPGA's programmable logic cells. The disadvantages of this method are more design complexity and relatively lower calculation speed. However, it can be realized with most of low-price FPGAs and can be adopted in the high-volume productions.

In order to have the adaptability to all types of FPGAs, two types of arithmetic circuits are designed with the both approaches mentioned above. The signal interfaces of them are identical, but the instructions can be realized are slightly different, the circuit designed with the second approach can only realize multi-cycle auxiliary multiply instruction.



Data Address Generator(DAG)
I0
M0
I1
M1
I2
M2
I3
M3
In-Ram Address
Out-Ram Address
Input Data Memory 2 *(1K * 16-Bit)
Output Data Memory 2 *(1K * 16-Bit)
Data
Data
GPR01
GPR00
GPR03
GPR02
General Propose Register Set
GPR11
GPR12
GPR15
GPR14
I/O Data
STOUT (8-bit)
IOOUT (16-bit)
STIN (8-bit)
IOIN (16-bit)
GR Set Destination & Sources Address
Direct Data
Data
Arithmetic Logic Unit (ALU) & Multiplier(Mult)
Control Code
Program Squencer
Program Code
Program Rom (1K * 27-Bit)

Figure 1. The diagram of proposed ASIP PE

The diagram of arithmetic unit designed with the second approach is shown in Fig.2. The core part of this circuit is a 17-bit carry look-ahead adder. With the assistance of the auxiliary multiply circuit, this adder can realize multiply calculation with multi-cycle operation. As pointed in Fig 2, this function is actually accomplished by a typical 16-bit 2's complement sequential multiplier with modified Booth's algorithm [2] to generate the partial products. A partial production generation instruction – ppgen is assigned to this function, and a 16-bit 2's complement multiplication can be accomplished by successive execute ppegn-instruction 10 times (the function of last ppgen-instruction is to load the 32-bit multiply results into the destination GPRs). The reason for adopting this circuit structure is that it can obtain the best optimization result of speed and circuit scale in the FPGA [7].

Four status labels generated from the calculated result are stored in the status register. They are 'Zero', 'Carry', 'Overflow' and 'Negative'. These labels provide the conditions to the conditional branch instruction.

2) General Purpose Register Set (GPRs)

The general purpose register set (GPRs) consists of sixteen 16-bit registers. The main functions of the GPRs are providing the operands to the arithmetic unit and storing the calculated results from them. It also functions as the data exchange 'bridge' over the arithmetic unit and the data memory. The GPRs is accessed by its name or register number. To the multiply instructions, the GPRs can be configured as eight 32-bit register pairs, so that the 32-bit multiply result can be stored simultaneously in same instruction cycle.

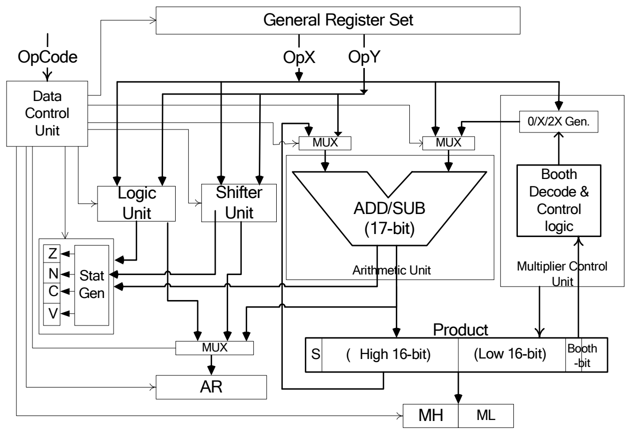


Figure 2. The diagram of the arithematic unit designed with the second approach

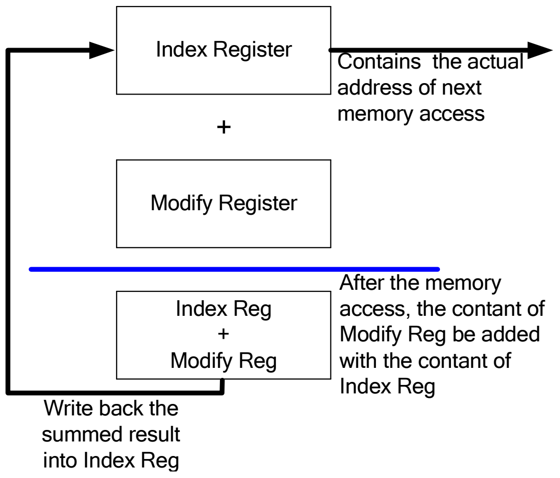

Index Register
Contains the actual address of next memory access
+
Modify Register
Index Reg + Modify Reg
After the memory access, the contant of Modify Reg be added with the contant of Index Reg
Write back the summed result into Index Reg

Figure3. Illustration of data memory addressing

3) The Data Address Generator (DAG)

The functions of data address generator are: addressing the operands from GPRs; addressing the destination address of calculated result to GPRs; and addressing the data memory address as data moving between GPRs and memories. The addressing of the GPRs is by the way of register addressing, and the addressing of the data memory is similar to the general CPU's index addressing, i.e, the actual address of the memory to be accessed is equal to the summation of the two values contained in the index register and modify register . This addressing mode is illustrated in Fig.3.

4) Program Sequencer

The program sequencer provides the program ROM address of the instruction to be executed. It controls the program execution of the ASIP processing element. The functions can be realized by it are shown as follows:

- Sequential execution of the program.
- Un-conditional / conditional branch of the program. Conditions: Four state labels stored in the status register and 'great', 'less', 'great equal', 'less equal' conditions generated from status register.
- Subroutine program call and return: 4-level of nested subroutines are permitted.
- Zero overhead looping: loop can be nested 2-level deep, the end-loop condition is the value of the loop counter (CE) minuses to zero. The initial value of the CE can be direct number in the instruction or comes from a GPR.
- Idle and wake up: provide a synchronization mechanism to the program execution and the external events. After the idle instruction is executed, the value of PC keeps unchanged, once an external event is arise, the value of PC is incremented, the program go on executing.

B. The Specific Instruction Set

The specific instruction set of proposed ASIP processing element is obtained by carefully researching of various kinds of engineering DSP programs [4]. The instruction set is shown in Table I. All the instruction can be accomplished in a single clock cycle.

This instruction set is universal to some extend, so it can be applied to many applications. In practice, it should be tailored according to the applied DSP program, so that the complexity of the PE circuit can be reduced more and the execution speed and the number of PE integrated in a single FPGA can be increased further. For the convenience of program debugging, a specific assembler program is developed. This assembler can output various types of debug files in the different development phase of the system. For example, in the step of behavioral simulation, it can output a program ROM's Verilog HDL behavioral description file, and in FPGA implementation step, it can output a coefficient file (COE) to load the executable machine code into FPGA's macro ROM.

The assembler has the self-contained functions such as syntax check and error detection. It can do statistics of the instructions used in the processing program by instruction type also. It can help the developer to optimize the specific instruction set and the circuit structure.

C. Memory Structure and Data Exchange Registers

The construction of memory structure relays on the system performance requirements and the input data structure. In order to speed up the I/O data transfer rate, two separate memory banks are used. One for the input data storage and the other for output data storage. Each memory bank can be constructed with two-page dual-port RAM, so that the whole processing work can be done with 3-stage pipelines: (input data storage) →(data processing & result storage) →(processed data output). The advantages of this memory structure are shown as follows:

- Data can be in/out of the PE memory using DMA mode with very high transfer speed.

TABLE I. SPECIFIC INSTRUCTION SET OF ASIP PE

Type	Instruction	Description
Logical	gr#d=gr#s1 and ge#s2	Logical and
	gr#d= gr#s1 or ge#s2	Logical or
	gr#d=gr#s1 xor gr#s2	Logical xor
	gr#d = not gr#s	Logical not
	clralu	Clear status reg
	tst gr#d bit#d	Bit test
Shift	gr#d = lrshift gr#s	Logical right shift
	gr#d = llshift gr#s	Logical left shift
	gr#d = arshift gr#s	Arith right shift
arith	gr#d = gr#s1 + gr#s2	add
	gr#d = gr#s1+gr#s2+c	Add with carry
	gr#d = gr#s1 - gr#s2	sub
	gr#d = gr#s1-gr#s2-c	Sub with borrow
	gr#d = ppgen	Partial product generation*
	gr#d = gr#s1 * gr#s2	Multiply**
Data Exchange	ld gr#d,gr#s	GPR data exchange
	ld gr#d,#data	Direct data load into GPR
	ld sr#d,gr#s	GPR load into SPR
	ld gr#d,sr#s	SPR load into GPR
	ld gr#d,address	Memory data load into GPR
	ld address,gr#s	GPR data load into memory
Program Control	rst	Soft reset
	idle	Program idle (wakeuped by external event)
	nop	nop
	jump addr	Jump directly
	j-on-c addr	Jump on condition
	call addr	Subroutine call
	rts	Return from sub
	do loop until ce	Loop control

- Can obtain adequate processing time.
- Suitable for real-time data processing systems.

All the memory accessing is done by data address generator (DAG). Each memory bank have two sets of data addressing register, one set for data write-into the memory, the other for data read-out off the memory.

When the multiple PEs are used to construct a parallel process-unit array system, as the key of the design, inter-PE data exchange problem must be solved. To solve the problem, a data exchange register set including data input/output registers, state and control signal input/output registers is added in the design. With the help of the data exchange register set and the data exchange matrix we have designed, each PE can communicate with its neighbor PEs efficiently [1, 8, 3]. This data exchange mechanism has been fully tested by a 1024-point FFT ASIP array which contains 16 PEs.

III. FPGA IMPLEMENTATION OF THE ASIP PE

The whole circuit of proposed ASIP PE is implemented into a Xilinx Virtex-V XC5VLX85-1FF676C FPGA to collect the statistic data of its important parameters such as circuit scale, running speed etc. The Device utilization summary report generated by Xilinx ISE 9.1i software is shown in Table II.

TABLE II. DEVICE UTILIZATION SUMMARY REPORT

Device Utilization Summary			
Slice Logic Utilization	***Used***	***Available***	***Utilization***
Number of Slice Registers	581	51,840	1%
Number of Slice LUTs	716	51,840	1%
Number used as logic	713	51,840	1%
Number of route-thrus	25	103,680	1%
Slice Logic Distribution			
Number of occupied Slices	354	12,960	2%
Number of bonded IOBs	108	442	24%
Number of* BlockRAM/FIFO	1	96	1%
Total Memory used (KB)**	36	3,456	1%
Total equivalent gate count for design	140,839		

The total equivalent gate count of the ASIP PE is 140,839, as shown in Table II. If the Macro cell used as RAM and ROM are excluded, this figure will be reduced greatly. Since all kinds of in-chip recourse utilization are below 2% except the IOBs, it is possible to integrate more than 36 ASIP PEs into this type of FPGA to construct a parallel processing system with ASIP PE array architecture. Through the careful analysis of the timing report and post simulation result, the instruction execution speed of a single PE can achieve at least 200MIPS. For the practical applications, the circuit will always be tailored, thus the circuit complexity will be further reduced, and the execution speed will be boosted also.

A median filter algorithm and a 5-3 lifting wavelet with soft threshold de-noise algorithm are programmed and applied to several 32×32-pixel images to test the function and performance of proposed ASIP PE. The processing results are listed in Table III. The total number of instructions of median filter algorithm is 92236, and the total number of instructions of 5-3 lifting wavelet is 78322.

A 1024-point FFT ASIP Array which contains 16 PEs is developed by us. All of the PEs are connected with the data exchange matrix as mentioned before, each PE can do 64-point FFT. Table IV shows the results.

The test results of practical applications mentioned above can prove that the ASIP processing element designed by us has certain advantages, such as simple circuit structure, self-contained instruction set, high FPGA in-chip resources utility and can be easily tailored to meet the needs of applications.

IV. CONCLUSION

A design of an ASIP processing element which is suitable for FPGA implementation with RISC like architecture is introduced in this paper. Take the designed processing element as the core process-unit, a parallel ASIP processing array system with high computational density can be implemented in a single FPGA with MIMD or SIMD architecture. The carefully designed circuit structure and specific instruction set of proposed processing element make it very universal, and thus can be tailored easily for many particular applications. This ASIP PE can also be used in the embedded applications. The development of proposed ASIP PE is simple, with the help of the specific assembler, many kinds of DSP algorithms can be realized by the way of software programming. Compared

TABLE III. IMAGE DE-NOISE RESULTS

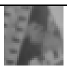
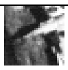
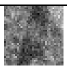
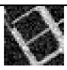
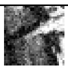
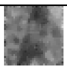
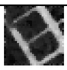
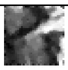
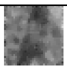
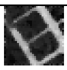
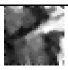
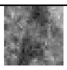
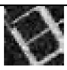
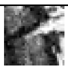
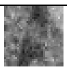
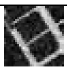
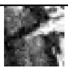
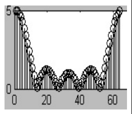
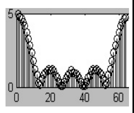
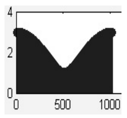
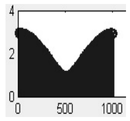
Original image			
Image with noise (PSNR/dB)	 18.4026	 20.8685	 22.3984
Median filter algorithm Matlab simulation Result (PSNR/dB)	 23.1013	 21.0724	 22.7739
Median filter algorithm ASIP processing result (PSNR/dB)	 23.1013	 21.0724	 22.7739
5-3 lifting wavelet algorithm Matlab simulation result (PSNR/dB)	 21.5185	 22.4200	 24.3424
5-3 lifting wavelet algorithm ASIP processing result (PSNR/dB)	 21.5016	 22.4193	 24.2753

TABLE IV. 64- AND 1024-POINT FFT RESULT

Type	Input signal	Matlab simulation result	ASIP result	Number of instructions
64-point FFT	$x(n)=\sigma(n)+\sigma(n-1)+\sigma(n-2)+\sigma(n-3)+\sigma(n-4)$			25830
1024-point FFT	$x(n)=\sigma(n)+2\sigma(n-1)$			66367

with the processing systems implemented in FPGAs based on the hardware state machine architecture, the architecture based on the ASIP PE can get more flexibility of algorithms, more design reuse ability and lower design complexity. Several test results from practical applications prove the advantages mentioned above of the proposed design.

REFERENCES

[1] H. M. Chang, M. H. Sunwoo, and T.-H. Cho, "Implementation of a SliM Array Processor," In Proceedings of the 10th International Parallel Processing Symposium, Washington, DC, USA, 1996. IEEE Computer Society. pp. 771–775,

[2] C.S. Wallace, "A Suggection for a Fast Multilier," *IEEE Transactionson Electronic Computers*, EC-13(1):14–17, 1964.

[3] R. E. Gonzalez. Xtensa , "A Configurable and Extensible Processor," *IEEE Micro*, 20(2), pp.60–70, 2000.

[4] M. Gschwind, "Instruction Set Selection for ASIP Design," In 7th International Workshop on Hardware/Software Codesign *(CODES'99)*, pp. 7–11, 1999.

[5] M. Jain, M. Balakrishnan, and A. Kumar, " ASIP Design Methodologies : Survey and Issues," In *Proceedings of the* Fourteenth International Conference on VLSI Design, pp. 76–81, Jan 2001.

[6] K. Keutzer, S. Malik, and R. Newton, "From ASIC to ASIP:The Next Design Discontinuity," In *IEEE International Conference on Computer Design*, pp. 301–304, January 2002.

[7] M.A.Thornton, J.D.Gaiche, and J.V.Lemieux, "Tradeoff Analysis of Integer Multiplier Circuits Implemented in FPGAs," In IEEE Pacific Rim Conference on Communications, Computers *and Signal Processing*, pp. 301–304, 1999.

[8] M. Okada, T. Hiramatsu, H. Nakajima, M. Ozone, K. Hirase, and S. Kimura, "A Reconfigurable Processor Based on ALU Array Architecture with Limitation on the Interconnection," In Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05), volume 04, pp. 152a, April 2005.

[9] Saponara, S., Fanucci, L., Marsi, S., Ramponi, G., Kammler, D. and E.M. Witte, "Application-Specific Instruction- Set Processor for Retinex-Like Image and Video Processing," IEEE Transactions on Circuits and Systems II: Express Briefs, 54(7):596 – 600, July 2007.