Prototype Design of DSP Processor using Multiplier Oriented Reconfiguration

Melvin Mathew Philips[1], **Vishnu V. S.**[2]

[1]Sree Buddha College of Engineering, Department of Electronics and Communication Engineering, Kerala University, India

[2]Professor, Sree Buddha College of Engineering, Department of Electronics and Communication Engineering, Kerala University, India

Abstract: ***Most embedded systems developments in recent years have been primarily based on hardware level reconfiguration. Reconfigurable architecture requires many processing elements and configuration switches and it increases complexity of the design, but at the same time improves flexibility. Reconfiguration in ALU design can significantly increase speed of operations. In order to achieve high through put rate in ALU, an efficient switching between elements and also a significant reduction in clock cycle is required. In this work a prototype design of a DSP processor with some improvements done on the multiplier based reconfiguration module architecture to increase the throughput rate. For data sensitive computations in digital signal processing the proposed reconfigurable ALU could be used. The task for the DSP operations will be shared into the proposed cores as per time and size demand.***

Keywords: ALU, Cores, DSP processor, Multiplier, Prototype, Reconfiguration

1. Introduction

Hybrid field programmable gate arrays (FPGAs) integrate capable embedded processor cores with a reprogrammable fabric, bringing together software and custom hardware in a manner that makes reconfigurable computing attractive beyond its traditional support base. A software-centric view, but one in which complex computation can be offloaded to custom hardware accelerators, has long interested the reconfigurable computing community, and hence such coupling has been explored in the past. The ALU which is responsible for all the basic computations can be modified by this concept. To accelerate the processing instead of using separate hardware accelerators, reconfigurations of ALU could be considered. A controller which controls this reconfigurable module also provides a way of improving processing speeds.

In order to process different signals in embedded system, various efficient algorithms, such as Fast Fourier Transform (FFT) and finite impulse response (FIR) have been developed. These algorithms are complex and characterized by data intensive computation. For these applications, two extreme approaches are used for their implementations: software running on a general purpose processor (GPP) and hardware in the form of application specific integrated circuit (ASIC). In general purpose processor, it is flexible enough to support various applications but they cannot provide sufficient performance to handle the complexity of the applications. In the case of ASIC, one can optimize the implementation in terms of power and performance but it turns out to be a dependent application, and its direct architecture-algorithm mapping restricts the range and flexibility of applicability.

A reconfigurable architecture can provide the advantages of two approaches. Such architecture has higher performance than general purpose processor and wider applicability than ASIC.

2. Existing Design

A hypothetical microprocessor might only include an arithmetic logic unit (ALU) and a control logic section. The ALU performs operations such as addition, subtraction, multiplication, division, shifting and logical operations such as AND, OR, XOR, NOT, NOT etc. Result of every operation is stored in accumulator and which can be driven through the output port or it can be saved in memory locations. The control logic retrieves instruction codes from program memory and initiates the sequence of operations required for the ALU to carry out the instruction. The intermediate results will is stored in register files or in data memory sections. A single operation code might affect many individual data paths, registers, and other elements of the processor.

Where as a DSP processor should contain MAC units for filtering operations. But in a software centric view, the instructions could be used to get the desired MAC functions. The existing design don’t have number of MAC units, instead it used different set of instructions to bring out the filtering operations.

3. Proposed Architecture

In order to satisfy the demand of data-intensive computation in digital signal processing, a novel 16bit reconfigurable DSP ALU associated with the processor is combined together with the normal DSP processor is proposed for bringing different familiar algorithms with circuit substrate. The architecture of the DSP processor which is proposed is shown in Fig. 3.1. The architecture consists of reconfigurable ALU (RALU) based DSP processor, General DSP processor, Data memories, Register banks, Program Memory and a main controller.

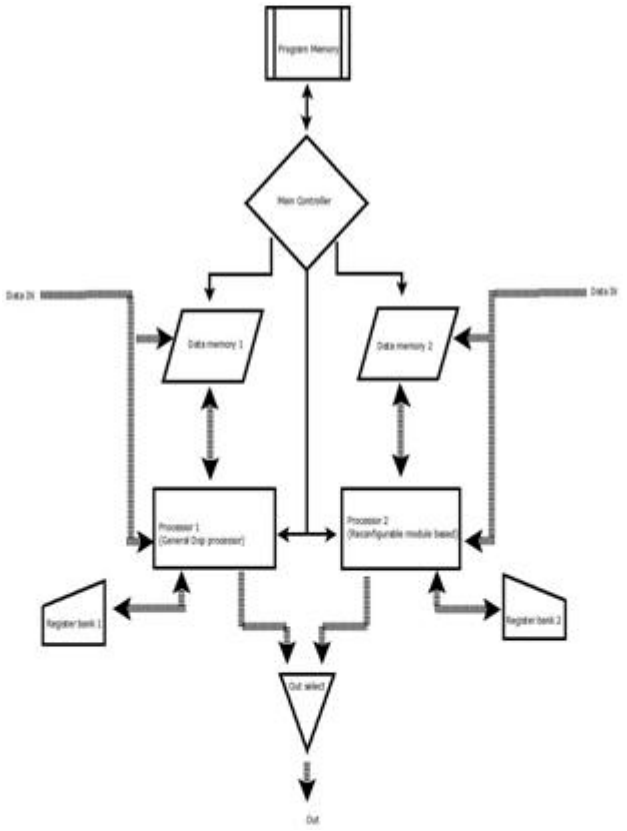

Figure 3.1: Proposed DSP processor architecture

3.1 General DSP processor

The block consists of an ALU which is basically designed for performing all the DSP operations. It consists of 40bit Adder-subtractor circuitary, 16bit multiplier combined together in an efficient manner to perform MAC operations. 40bit barrel shifter is used for performing shifting operations. 40 bit accumulator used for saving temporary values.

3.2 Reconfigurable DSP processor

This block consists of a multiplier based reconfigurable module used for getting adder, subtractor and multiplier output trough a single functional module. The ciruitary is actually based on Baugh Wooley design. By using different combinations, desired arithmetic functions are derived. The block itself has a 16bit accumulator for saving temporary values.

3.3 Data Memories

For each processor separate data memories are used. Each memory consists 2*512 space of 16bit data storage. Total data memory size is 2048*16 bit. The data memory address space is of 2048 bits. Each address can be incremented if the operation requires so.

3.4 Register Banks

Two register banks are allocated for both processors. Each bank consists of four registers of 16bit size. The intermediate results will be stored and could be called as per the operations.

3.5 Program Memory

The program memory is of size 512*16 bits. Instructions for any DSP operations are stored in the memory. The memory is controlled by the main controller. For each clock cycle, the instructions will be loaded into the processors as per the demand of the operation.

3.6 Main Controller

It controls the switching between processes which has to be fed to the processors. If one is busy with one operation, the controller switches the pending operations to the other. If the operations requires data processed in a specific manner, which also can be controlled by the controller. E.g. If the data values are expressed in signed types, then the operation will be directed to the second processor which is efficient in handling signed number arithmetic.

4. Operation

For checking the operation, the instructions for a fifth order FIR filter is stored in the program memory. Samples of 20 consecutive input values are stored in each data memories. While the first unit performs the operation in swiftly with the compromise of area, and energy constraints, the other performs slower. But the area utilization for the second processor is less compared to the first one. The energy and area efficiency is due to the reconfigurable architecture present in the second processor. The operations result is shown in Fig.4.1. Some of the required commands for the operations are also listed in table 4.1.

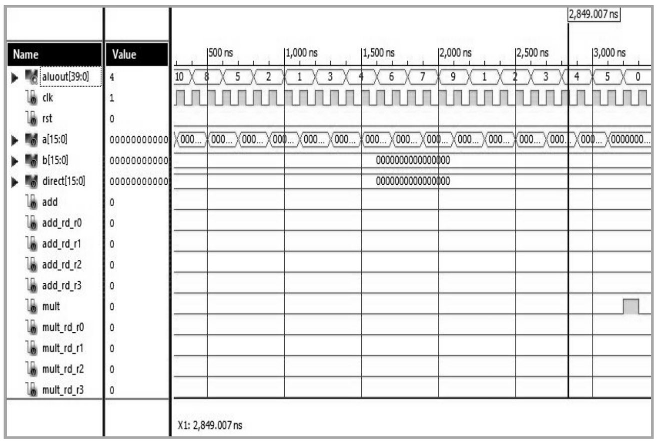

Figure 4.1: Output of FIR filter operation.

Table 4.1: Commands

Commands	*Description*
Add	Adds the value from both memory blocks and stores it in accumulator, increments the data pointers.
Add_rd_rX	Adds the value in register x with the direct value and stores in accumulator
Mult	Multiplies the values from both memory blocks and stores in accumulator, increments the data pointers.
Mult_rd_rX	Multiplies the value in register x with the direct value and stores in accumulator

MAC	Multiplies both the values from memory blocks and adds with accumulator value and stores in the accumulator
MAC_rd_rX	Multiplies direct value with register value and adds it with the current accumulator value and stores in the accumulator.
Reg_w_rX	Writes the direct value into the register X.
Store_acc_rX	Stores the accumulator values into register X.
Store_IN_rX	Stores the data IN value into the register X
Into_acc	Stores the data IN value in to the accumulator
Into_acc_rX	Stores the register X value into the accumulator
Mem_jmp	Jump to the memory locations addressed by the pointers
Mem1_jmp	Jumps to the memory location addressed by the pointer in data memory 1
Mem2_jmp	Jumps to the memory location addressed by the pointer in data memory 2
Mem1_write	Writes to the memory location of memory 1 pointed by memory pointer
Mem2_write	Writes to the memory location of memory2 pointed by memory pointer

5. Conclusions

Proposed prototype design of the DSP processor could be used for the implementation of efficient processors for DSP operations. The FIR operation performed for an ECG signal showed that the switching between the processors will result in significant area and power reduction. The number of clock cycles requires for each of the processors is different and it can be used for any optimization that could be performed in future. The design is done on Xilinx platform design suite 14.2 using Verilog tool.

References

[1] Manuel de la Guia Solaz, Wei Han, Richard Conway, "A Flexible Low Power DSP With a programmable Truncated Multiplier," IEEE Transactions on Circuits and Systems-I: Regular Papers, Vol.59. No.11, Nov 2012.

[2] Yoonjin KIM, Rabi.N.Mahapatr, "Dynamically Compressible Context Architecture For Lower Coarse Grained Reconfigurable Array, in Computer Design. 2007. 25th International Conference on, Oct 2007.

[3] Yoonjin KIM, Rabi.N.Mahapatr, "Dynamaic Context Compression for Low Power Coarse-grained Reconfigurable Architecture," In Very Large Scale Integration(VLSI) systems, IEEE transactions on DEC 2009.(Vol 18, Issue:1)

[4] Sheetal U. Bhandari, Shaila Subbaraman, Shashank Pujari, Rashmi Mahajan" Internal dynamic partial reconfiguration for real time signal processing on FPGA", in Indian Journal of Science and Technology Vol. 3 No. 4 (Apr. 2010).

[5] Callahan T.J, Hauser, J.R, Wawrzynek J, "The Garp Architecture and C Compiler", in Proc. IEEE Symp FPGAs for Custom Computing Machines on Aug 2002.

[6] Takashi Miyamori, Kunle Olukotun, "REMARC: Reconfigurable Multimedia Array Coprocessor", IEICE Trans. on Information System, 1999, pp.389-39.

[7] Hartej Singh, Ming-hau Lee, "MorphoSys: An integrated reconfigurable system for data-parallel and computation-intensive applications", IEEE Trans. on Computer, 2000, pp.465-481.

[8] Takayuki Sugawara, Keisuke Ide, Tomoyoshi Sato, "Dynamically reconfigurable processor implemented with IPFlex's DAPDNA technology", IEICE Trans. Inf. & Sys., 2004, pp.1997-2003.

[9] H. J. Oh, S. M. Mueller, C. Jacobi, "A fully-pipelined single-precision floating-point unit in the synergistic processor element of a CELL processor", IEEE J. Solid-State Circuits, 2006, pp.759-771.

[10] Chunyang Feng, Liang Yang, "Design and Evaluation of A Novel Reconfigurable ALU Based on FPGA", 2013 International Conference on Mechatronic Sciences, Electric Engineering and Computer (MEC) Dec 20-22, 2013, Shenyang, China