Computer Aided Design for Low Power Fir Processor on System On-Chip Platform Architecture for High Performance DSP Applications

A. Hemalatha[1], A. Shanmugam[2]

[1]Research Scholar , Anna University ,Chennai
[2]Principal, Bannari Amman Institute of Technology, Sathyamangalam

Abstract
The continually increasing integration density of integrated circuit, with astronomical increase in fabrication cost and enormous time to market portrays important paradigm shift in next generation System –on-Chip (SoC) design. This leads more programmable designs that can spin a wide range of applications. Hence there is a trend away from the fixed SoC to highly flexible SoC with improved time to market. The reconfigurability in SoC can be achieved either by the Programmable gate arrays [FPGA] and/or through Programmable interconnect. The emergence of static memory based FPGA that are capable of being dynamically reconfigured i.e. partially programmable during run time has been a driving force for flexible architecture. At the same time, the standardization of the intellectual property [IP] deliverables and their wide availability has started gaining importance towards expanding the possibility of reconfiguration in SoC Synthesis of SoC has become less complicated shortening design time, with the IP reuse which could be achieved by socketisation. The pre-designed and pre-verified IP blocks are obtained from the internal sources or third parties and combined onto a single chip. This arises a need of rapid integration of communication between different modules embedded in the system. The proposed design provides a solution for achieved with minimal cost of hardware and software, and improve the performance. In a Hardware design, the hardware complexity of the FIR filter is directly proportional to the tap length and the bit-width of input signal. To reduce the hardware cost, this can be figured out with iteration calculations by software; therefore, a co-design of hardware and software may produce cost-efficient FIR filters. The key design concept is to build a processor for software processing with minimum hardware resources, without sacrificing the performance of original FIR filter. The proposed design methodology can be considered as an intellectual property (IP) design for FIR filters in system-on-a- chip (SOC) environment.

Introduction

In recent years, more and more products have started out using DSP processors, fueling demand for faster, and smaller, cheaper and more energy-efficient integrated chips. These smaller, cheaper and more energy-efficient integrated chips open the door for a new wave of products to implement signal- processing capabilities. Although fundamentally related, DSP processors are importantly different from general- purpose processors (GPPs) like the Intel Pentium. To realize why, we need to know what is involved in signal processing. Some of the most common functions executed in the digital domain are signal filtering, convolution and fast Fourier transform. In mathematical terms, these most functions perform a series of dot products. This makes for us to the most popular operation in DSP: the multiply and accumulate (MAC).
The major architectural change that discerned DSP processors from the early GPPs was the addition of specialized hardware that enabled single-cycle multiplication. DSP architects also added accumulator registers to hold the summation of several multiplication products. Accumulator registers are typically wider than other registers, often providing extra bits, called guard bits, to avoid overflow. Another highly visible difference between DSP processors and GPPs lies in their memory structure. Typical DSP algorithms require more memory bandwidth than the Von Neumann architecture used in GPPs. Thus, most DSP processors use some forms of Harvard architecture which has two separate memory spaces, typically partitioned as program and data memories.
Although, this may seem that DSP applications must pay careful attention to numeric accuracy - which is much easier to do with a floating-point data path, fixed-point machines tend to be cheaper (and faster) than comparable floating-point machines. To maintain accuracy without the complexity of a floating-point data path, DSP processors usually include, in both the instruction set and underlying hardware, good support for saturation arithmetic, rounding, and shifting.Currently designs are considered at the system level.e.g. a module within a design itself represents a highly complex component. System Level Integration (SLI) is encouraging the trade of complex module designs so called, Intellectual Property (IP) between all types of providers and end users. This research aims to develop a

Manuscript received July 5, 2011
Manuscript revised July 20, 2011

scheme for implementation of low power high performance Digital Signal Processing intensive System-on-Chip platforms.
This research will provide an overview of interface strategy by which suggests dynamic plugging of various IP blocks, with the aid of programmable interconnect to enable ever changing system that can be adapted to almost any requirement for SoC, Reconfigurable Pipeline Data paths [RaPiD]. Simulations will be carried out with an FIR filtering on to the platform. Results will be provided for timing and power consumption of the whole SoC platform. This research will involve synthesis and simulation of complete systems with multiple high performances DSP IP cores which include blocks for audio and image manipulation together with IP cores for transmitting these over a range of channels.

Trends in the DSP Processor Architecture

The fundamental difference between a DSP processor and a generic processor is the DSP Processor's hardware multiplies-accumulate (MAC) block and specialized memory and bus structures to facilitate frequent data access commonly found in DSP applications. The MAC operation is usually the performance bottleneck in most DSP applications. DSP processor vendors incorporate MAC blocks in their architecture to minimize this performance bottleneck. Some DSP processor vendors have also tried adding multiple MAC blocks to their architecture to boost the overall multiplier bandwidth. For example, the TMS320C6411 device from Texas Instruments can calculate up to eight 8 × 8-multiplication results in a single clock cycle. While adding more MAC units may provide more DSP throughput, the processor falls behind in raw data processing power for certain data-intensive DSP functions such as Viterbi encoder/decoder and FIR filters. To work around this problem, DSP processor vendors have also tried incorporating a hardware accelerator coprocessor) block such as the Viterbi coprocessor, turbo coprocessor and the enhanced filter coprocessor. While such coprocessor blocks provide high DSP throughput, they do not cater to all DSP applications. Most DSP applications cannot benefit from the DSP vendors' predefined hardware accelerator blocks. Additionally, such hardware accelerator blocks are fixed, do not allow for any level of customization for the specific design needs, and can quickly become obsolete in todays.

Efficient CPU Architecture for DSP Processors

The core unit of the FIR-Processor is the embedded CPU, which is an eight bit "Von Neumann" machine with single accumulator. It has an eight-bit data bus; six-bit address bus; neither stack nor status flags are present (Fig. 1). The instruction set contains opcodes: conditional branch, load memory into accumulator, store accumulator into memory, and add accumulator with memory.

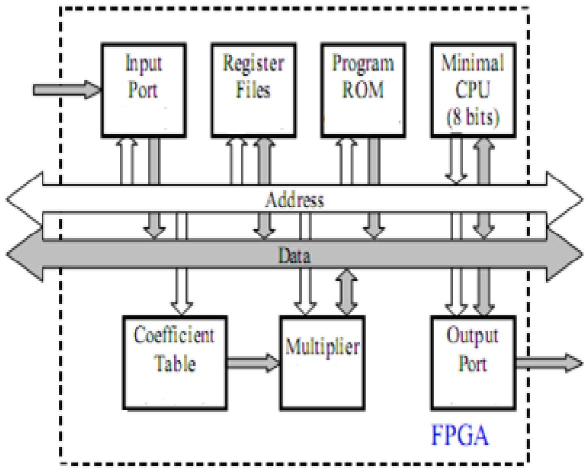

Figure 1:FIR Processor

A suggested instruction set

Fixed-point DSP processor instruction sets are designed with two goals in mind. They must:
• Enable the processor to perform multiple operations per instruction cycle, thus increasing per-cycle computational efficiency, and
• Minimize the amount of memory space required to store DSP programs.
To accomplish these goals, DSP processor instruction sets generally allow programmers to specify several parallel operations in a single instruction. However, to keep the word size small, the instructions only permit the use of certain registers for certain operations and do not allow arbitrary combinations of operations. The net result is that DSP processors tend to have highly specialized, complicated, and irregular instruction sets. In our approach, CPU architecture for a DSP processor, related instruction set generally includes 7 categories of operations.

Arithmetic and logic unit (ALU)

The 8 bit ALU, shown in Figure 2, implements a wide range of arithmetic, logical, and rotate functions. The result is transferred to a destination accumulator (A or B) or sent to the output bus for memory writing. There are two input buses to read words of memory in a cycle and write the result to the memory.

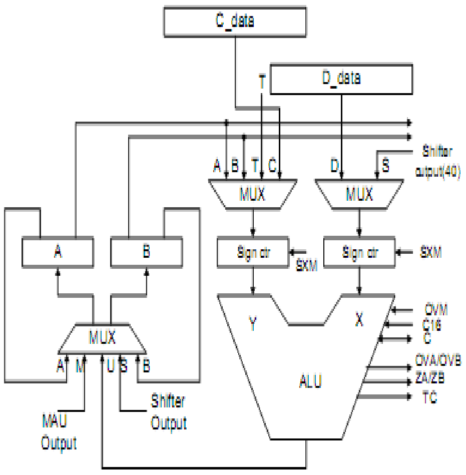

Figure 2: Arithmetic Logic unit

ALU input takes several forms from several sources. The X-input source to the ALU is either
of the two values:
• The shifter output (shifted 8-bit data-memory operand or a shifted accumulator value).
• A data-memory operand from data bus (D_data). The Y-input source to the ALU is any of three values:
• The value in one of the accumulators (A or B).
• A data-memory operand from data bus (C_data).
• The value in T registers.

Multiplier and adder unit

The CPU architecture has a 8-bit ×8-bit hardware multiplier coupled to a 16-bit dedicated adder. This multiplier/adder provides multiply and accumulates (MAC) capability in one cycle. The multiplier can perform signed, unsigned, and signed/unsigned multiplication. The multiplier output can be shifted left by one bit to compensate for the extra sign bit generated by multiplying two 16-bit 2s-complement numbers in fractional mode. The adder's inputs come from the multiplier's output and from one of the accumulators. Once any multiply operation is performed in the unit, the result is transferred to a destination accumulator (A or B). This structure also has been designed to determine absolute value of X-input when Y-input is fed through constant one value.Multiply-accumulate instructions such as:MAC Xmem, Ymem, src[,dst],which does the following operation: (Xmem) × (Ymem) + src → dst
All of these operations should be performed in just one cycle. Dot product that is commonly used in DSP applications can be executed with MAC instruction embedded in a repeat instruction.

Registers and memory

The CPU is accumulator based and supports minimal registers. The ALU is 8 eight bits wide without carrying bit generation as the overflow flag. The PC has a width of six bits, which allows addressing 64 bytes of memory. The memory space is shared between program code, data and I/O devices (tab. 1).

Table :1 Rigister and memory

Address	***Purpose***
00-1F	Program memory
20-2F	External memory or devices
30-33	Constants of coefficient
38	X(n) input register
39	Y(n) output register
3A	Result of multiplication
3B	Reserved
3C	X(n-3) register
3D	X(n-2) register
3E	X(n-1) register
3F	Temporary A(n) register

Execution of instructions

To achieve an efficient design for CPU architecture, we had to clearly manifest every aspect of each instruction in its hardware implementation. We required discovering optimized arithmetic and logical operations enough to execute all instructions. In addition, we had to determine which component is the best for executing each instruction.Fixed-point DSP processor instruction sets are designed with two goals in mind. They must:
• Enable the processor to perform multiple operations per instruction cycle, thus increasing per-cycle computational efficiency, and• Minimize the amount of memory space required to store DSP programs.To achieve these goals, DSP processor instruction sets generally allow developers to specify several parallel operations in a single instruction. However, to keep the word size small, the instructions only allow the use of certain registers for certain operations and do not allow arbitrary combinations of operations. The net result is that DSP processors tend to have highly specialized, complicated, and irregular instruction sets.

Table: 2 Instructions

Mnemonic	***Description***
LD Smem [,SHIFT], dst,	Load and store instructions
ADD Xmem, Ymem, dst,	Arithmetic instructions
XORM #lk, Smem,	Logical instructions
ST src, Ymem || ADD Xmem, dst,	Arithmetic instructions with parallel store and load
MAC Xmem, Ymem, src[,dst],	Multiply-accumulate instructions

Filter Implementation Structure

FIR (Finite Impulse Response) filters and IIR (Infinite Impulse Response) filters are two primary types of digital filters used in digital signal processing applications. In comparison, FIR filters require more memory and/or calculation to achieve a given filter response characteristic, but they offer some advantages, such as:

• They have "phase linearity" characteristic. The output signal is delayed, but its phase is not distorted.

• They are simple to implement by using "finite-precision" arithmetic and fractional arithmetic.

• They can use coefficients less than 1.0 in magnitude and simplify the implementation with fixed-point processors.

• They are suitable for multi-rate applications by "decimation", "interpolation" or both.

Finite Impulse Response (FIR) IP Cores are important blocks in both audio and video signal processing. In digital systems, noise reduction, echo cancellation etc are repetitively executed with the help of FIR filters.The main objective of this work is to present a design methodology for an FIR Filtering IP Core that is parameterized and programmable. The sequential implementation is selected to minimize the overheads of design. The proposed architecture has capability of run time programmability for SoC design. Type of filter, number of coefficients, word length for input data and filter coefficients can be changed. The objective of the following simulations is to experiment with a 4-tap FIR filter built with the designed processor and implemented with sequential method saving hardware resources. The input and output signals to and from the filter are the unsigned 8-bit x [7..0] and the unsigned 8-bit y [7..0]. This filter implements the following FIR equation:

$y(n) = x(n)h(n) + x(n-1)h(n-1) + x(n-2)h(n-2) + x(n-3)h(n-3)$ (1)

where $h(n) = 7/16$, $h(n-1) = 5/16$, $h(n-2) = 3/16$, $h(n-3) = 1/16$.

This equation can be rewritten as:

$y(n) = 1/16 * 7x(n) + 1/16 *5x(n-1) +1/16 * 3x(n-2) + 1/16 *1x(n-3)$ (4.1)

or

$y(n) = 1/16 * (7x(n) + 5x(n-1) + 3x(n-2) + 1x(n-3))$ (2)

The equation 2 sums all multiply products then divided by 16, this will make an overflow calculation on the 8 bits ALU, therefore each multiply product should divided by 16 (i.e. shifts 4 bits to right) first and sums later for the FPGA design.

SYNTHESIZES AND PERFORMANCES

We have analyzed area and speed of di□erent FIR filters architecture, using chip resources of four versions of hardware implementations are listed in table 3. Simulations were performed for generated data samples using modelsim simulator. This was followed by computing their area and speed with quatrus II EDA tool. In all cases, a clock rate of 25MHz and 1.8V were assumed.The 8- bits unsigned multiplier can be synthesized either by logic element or by dedicated multiplier circuit (V1.1 and V1.2). It occupies 34 combinational logic cells or one 9-bits DSP block respectively. To compare the performance, a pure FIR hardware circuit was synthesized with the same type FPGA chip (V2.0). Although the version 1.1 and 1.2 uses the most combinational logic cells (LC), its multiplier is in the LCs, this made the design synthesis suitable for FPGA chips without internal DSP blocks.

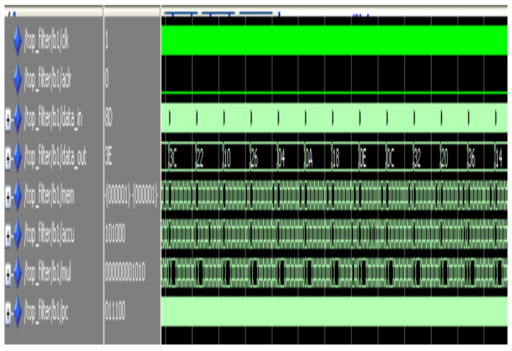

Figure 3: simulation result

Table 3: Hardware implementation result

Implementation	LC Combinational	LC Registers	9 bits DSP Block	Total logic element	Fmax (Mhz)
V1.1	145	71	0	171	79.62
V1.2	123	71	1	141	84.34
V2.0	77	57	0	102	123.46
V3.0	29	18	1	35	364.96

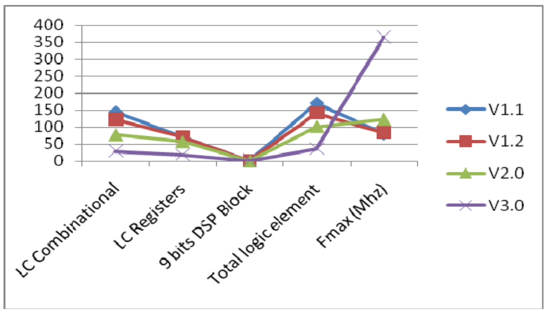

Figure 4: comparison chart for various processors

CONCLUSIONS

In this paper a novel design methodology of FIR-Processor on system on chip platform for DSP application is presented. We alleviate the cost-efficient high speed chip implementation for SoC systems, which offers some advantages over conventional methods, such as:

- It can be synthesized and fitted to a FPGA/CPLD; many identical units can be integrated into one chip for parallel processing.
- It has the flexibility of easy changing values of coefficients, easy extension to n-tap FIR filter, and easy
- Expansion to outside word using internal tri-state I/O bus.
- Enable the processor to perform multiple operations per instruction cycle, thus increasing per-cycle computational efficiency, and
- Minimize the amount of memory space required to store DSP programs.

The parameterized cores are an essential component for IP based SoC design.

Implementation were carried out with an FIR filtering IP core integrated on to the platform. Results were provided for timing and area consumption of the whole SoC platform.

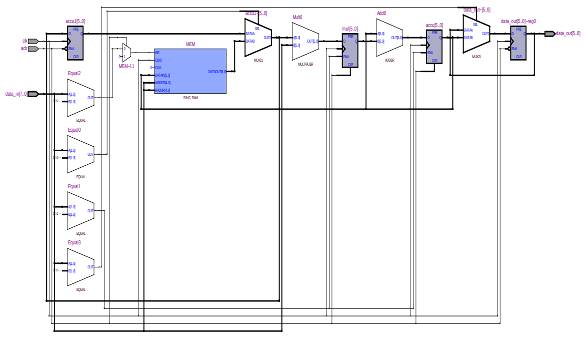

Figure 5:RTL Schematic for FIR Processor

REFERENCES

[1] C.H.Wang, A.T.Erdogan, T.Arslan, "High throughput and low power FIR filtering IP cares", Proceedings of IEEE International SOC Conference 2004, pp. 127 –130, Sep. 2004.

[2] M.B.I.Reaz, M.T.Islam, M.S.Sulaiman, M.A.M.Ali, H.Sarwar, S.Rafique, "FPGA realization of multipurpose FIR filter", Proceedings of the Fourth International Conference on Parallel and Distributed Computing, Applications and Technologies, PDCAT'2003, pp. 912 –915,Aug. 2003.

[3] A.T.Erdogan, M.Hasan, T.Arslan, "A low power FIR filtering core", Proceedings of 14th Annual IEEE International ASIC/SOC Conference 2001, pp. 271 –275, Sep. 2001.

[4] A.T.Erdogan, T.Arslan, "High throughput FIR filter design for low power SoC applications", Proceedings of 13th Annual IEEE International ASIC/SOC Conference 2000, pp. 374 –378, Sep. 2000.

[5] J.Valls, M.M.Peiro, T.Sansaloni, E.Boemo, "Design and FPGA implementation of digit-serial FIR filters", 1998 IEEE International Conference on Electronics, Circuits and Systems, Vol. 2, pp. 191 –194, Sep. 1998.

[6] Lee Hanho, G.E.Sobelman, "FPGA-based FIR filters using digit-serial arithmetic", Proceedings of Tenth Annual IEEE International ASIC Conference and Exhibit 1997, pp. 225 –228, Sep. 1997.

[7] HUANG, J.R., IYER, M.K., CHENG, K.T.: 'A self-test methodology for IP cores in bus- based programmable SoCs', VLSI Test Symposium, 19th IEEE Proceedings on VTS 2001, 2001 pp. 198 –203.

[8] WILTON, S.J.E., SALEH, R.: 'Programmable logic IP cores in SoC design: opportunities and challenges', Custom Integrated Circuits, 2001, IEEE Conference on 2001 pp. 63 –66.

[9] KIM, K.W., KWANG, H.B., SHANBHAG, N., LIU, C.L., KANG S.M.: 'Coupling-driven signal encoding scheme for low-power interface design', Computer Aided Design, 2000. ICCAD-2000. IEEE/ACM International Conference on, 2000, pp. 318 -321.

[10] YOO, S.J.; NICOLESCU, G., LYONNARD, D., BAGHDADI, A., JERRAYA, A.A.: 'A generic wrapper architecture for multi-processor SoC cosimulation and design', Hardware/Software Codesign, 2001. CODES 2001. Proceedings of the Ninth International Symposium on, 2001 pp. 195 -200.