menu

"Tradeoffs in Multiply Accumulate Architectures for Efficient Implementation of DSP Algorithms"

Neal S. Stollon neal@itc-usa.com
Christopher L. Connell chris@itc-usa.com
George Landers george@itc-usa.com
Infinite Technology Corporation
Richardson, Texas

Abstract: Whereas previous generations of DSP silicon were limited to one or 2 simple MACs per chip at most, emerging application specific DSP architectures may throw larger numbers of more sophisticated MACs at multiply-accumulate intensive applications to increase throughput and performance. In this paper, some approaches to MAC-centric DSP system architectures and chip design that evolve when "MACs become cheap" are presented. Architectural and memory integration issues that should be considered in MAC intensive designs, including design tradeoffs of different multiply accumulate structures for high performance, silicon area tradeoffs, and efficient implementation of common DSP algorithms are discussed.

1. Introduction

Digital Signal Processing (DSP) algorithms require multiply-accumulate (MAC) intensive resources for most operations. The efficiency and performance of the MAC resources are often the driving feature in the throughput and effectiveness of DSP applications. The emergence of System on Chip (SoC) densities of computing resources, with gate counts in the range of millions of gates, along with IP sources that can provide pre-configured blocks of high performance logic tailored for applications, have the potential to change the way that DSP architectures are implemented. Whereas previous generations of design have been based on the premise that hardware is expensive and that complex implementation and shared resources were necessary, current generation designs have the option of multiple instances of blocks and dedicated supporting resources to accelerate performance. The differing configurations of multiply and accumulator elements that may be made in implementing MAC block architectures are examples of the tradeoffs that potential increases in SoC resources allow. Among the types of global tradeoffs that might be addressed in MAC design are:

The type of multiplier and level of pipelining based on performance requirements

- The amount of dedicated RAM resources (both depth and number of ports) allocated to the multiply and accumulator to reduce "data starved" operation
- The accumulator size, which allows larger numbers of recursive "inner loop" operations to be summed without saturation or overflow
- Accumulator complexity and features

Custom MAC architectures may be considered when high performance complex multiply operations are required for a given application with higher performance requirements than those supported by standard DSP. Since MAC architecture is typically the limiting factor in FFT and other transform based algorithms' performance, the ability to address multi-channel or multi-stage algorithms on a single chip is often a direct function of the number and performance of MAC resources that are available. One critical measure of the performance of MAC architectures in DSP inner loop operations is the proportion of time that MAC operations are occurring, as opposed to the MAC idling while waiting for data to be routed or loaded.

The performance impact of integrating dedicated local memory, for both scratchpad and coefficient storage with MAC blocks is a typical configuration tradeoff that can improve the throughput of traditional MAC structures and multi-element MAC architectures. Memory access is a factor that is often under-scoped in the design of MAC architectures; many MAC intensive applications where multiple MAC operations are required do not make full use of their MAC resources due to data starvation caused by a lack of dedicated memory resources. MAC structures that allow local memory integration can achieve near capacity throughput levels (which can be up to 10x greater than scalar DSP processor architectures where computing resources share common memory) with corresponding area penalty of less than 2X.

The flexibility of merging MAC and local memory block architectures compliments new classes of reconfigurable signal processing architectures that allow MAC intensive datapaths to be dynamically configured for optimal performance. Infinite Technology, as a vendor of datapath hardware IP, makes use of several variants of complex MACs to improve the throughput of inner loop DSP algorithms. MAC intensive inner loop DSP functions can benefit from enhancements in complex multiply and accumulator block architectures, localized memory support of MAC operations, and MAC functional reconfiguration

include high throughput communications areas such as spectral analysis and imaging functions, including block transform and decimation.

2. MAC throughput and pipelining issues

A typical MAC function can be simply represented as a 2 stage pipelined design using cascaded multiplier and accumulator blocks. As such, maximum throughput is achieved for simple sum of product applications, such as FIR filtering, where data can be supplied at a constant and regular rate. Due to the complexity of the multiplication function, it is typically the pipeline stage that defines MAC performance.

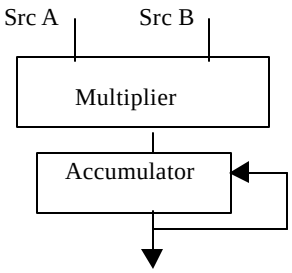

Figure 1: MAC Function

Improved multiplier block performance has been the subject of significant ongoing work, which has resulted in the development of a range of approaches, (bit serial/ parallel, digit re-coding over different radix (Booth algorithm), array based (Wallace Tree), partial product and shift-carry based architectures). These differing multiplier approaches allow the designer to tradeoff performance vs. pipeline staging vs. gates vs. elegance and complexity [1]. While space precludes any detailed discussion of multiplier architecture issues, for purpose of this paper, we will assume, without loss of generality, a Wallace Tree architecture for discussion purposes as a general purpose multiplier block.

A common DSP operation is the complex product over 2 sets of real and imaginary inputs A and B which are complex numbers (Ai,Aj) and (Bi,Bj) and where

$$C = A * B = (Ai + Aj) * (Bi + Bj).$$

The complex resultant CiCj is formed as

$$Ci = (Ai * Bi - Aj * Bj) \text{ and}$$
$$Cj = (Ai * Bj + Aj * Bi)$$

The generation of complex products and related vector multiplication operations are the primary inner loop functions of a number of DSP applications, the FFT being perhaps the most well known, if not most widely used. Streamlining of the complex multiply algorithm for improved performance can be addressed by two modifications to the MAC structure; incorporation of multiplex registers for staging of operands used in the complex multiply operation and the addition of a second accumulator stage that allows interleaved accumulation of partial product terms.

Complex multiply operations generate 2 partial products (real and imaginary), that constrain a single accumulator based MAC, and require the availability of 4 operands, each of which are used twice in 4 clock cycles in the optimum single multiplier throughput case. MAC performance in complex multiply applications can be improved and simplified by the addition of dedicated register files for pre-loading operand data and a second accumulator stage to address the summation of the second partial product of the complex multiply. A block diagram of such a Single Multiplier/Multi-Accumulator (SM-MAC) Execution Unit is shown in Figure 2.

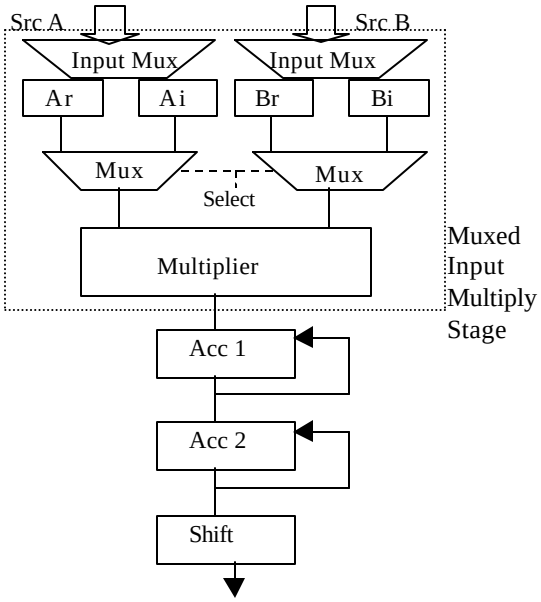

Figure 2. Single Multiplier/Multi-Accumulator (SM-MAC) Execution Unit

The single multiplier and the standard 1st accumulator followed by a 2nd accumulator can complete a full complex multiply in four (4) cycles without extraneous data manipulation. four muxed registers are merged with the multiplier stage to capture operand data from system busses in two clock cycles, i.e.: bring in both real partial products (Ai, Bi) on the 1st cycle and then both imaginary partial products (Aj, Bj) on the 2nd cycle.

On the 2nd cycle, real operands have been loaded for the Ai * Bi multiply operation and placement of the real partial product into the 1st accumulator. On the 3rd cycle, both imaginary operands have been loaded, and Ai * Bj are multiplied and the imaginary partial product is placed into the 2nd accumulator On the 4th cycle, Ai * Bj are multiplied and added to content of the 2nd accumulator. On the 5th cycle, (whose operation is the same as the 1st cycle) Aj * Bj are multiplied and added to contents of the 1st (real

data) accumulator. The complex multiply operation is therefore completed in 4 cycles by the SM-MA Execution Unit, with real and imaginary sum of products produced every other cycle.

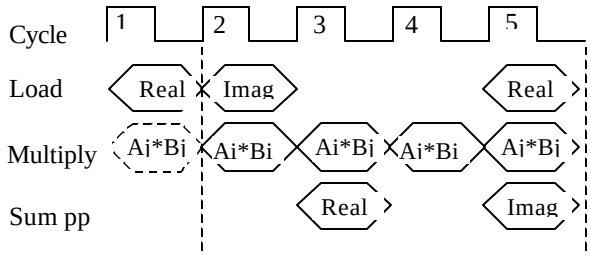

Figure3: A 4 Cycle Complex Multiply Operation using SM-MAC

The SM-MAC also incorporates an output shifter to support data scaling to be performed for the conditions of normalization and proportional division.

3. Dual MAC structure for DSP arithmetic

An increasingly recognized issue in submicron SoC design is that large numbers of independently connected computing elements (MACs, ALUs, Memory, etc.) in standard DSP architectures are complex to control and expensive in terms of silicon area for interconnection due to instruction word size limitations. With decreasing gate geometry making chip physical size increasingly interconnect limited, the number of global data busses available intra-chip may be limited. As an option to treating individual MACs as standalone execution units, the efficiency of DSP operations can be increased by merging individual MACs and their supporting resources into multi-MAC execution units. This use of such an execution unit structure has additional advantages of being able to package a complete verified function as IP with higher relative value than a set of standalone arithmetic blocks.

Figure 4 shows a Multiple MAC Execution Unit that incorporates dual multiplier blocks. In order to support arithmetic operations on the two intermediate generation products or the MACs, the accumulator stage is replaced with a general purpose ALU of equivalent accumulator length for complex arithmetic computing. By replacing the accumulator with a more robust add/subtract ALU, the execution unit is able to support diverse extended post-multiplication manipulation of data, including absolute value, comparison, and implementation of both MAC and MAS (multiply and subtract) functions.

An additional block replacement is the shift function used in the previous MAC with a more general data selector. The Selector block included algorithm specific features that can be used, in addition to data shifting operations, to select desired product words (high or low) and sign or saturate the result as appropriate before routing the data to subsequent staged operations.

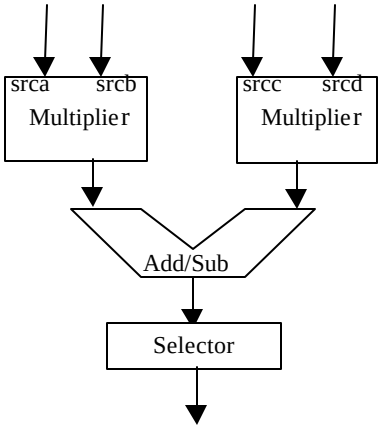

Figure 4: A Half Complex Multiply (HCM) MAC Execution Unit

The Dual MAC architecture in Figure 4 implements a Half Complex Multiply (HCM) function. The HCM MAC execution unit does a complex multiply in three clock cycles provided it can receive operand data at that rate. During FFT execution all memory ports are active 100% of the time. I/O cannot be hidden behind processing. A typical complex multiply engine using this architecture would integrate up to six ports of buffered or dedicated memory, to provide a level of local elastic storage between the HCM and large memory blocks. Local elastic storage helps to ensure full throughput HCM operation under conditions of burst mode memory update or multiple access conflicts at the system level.

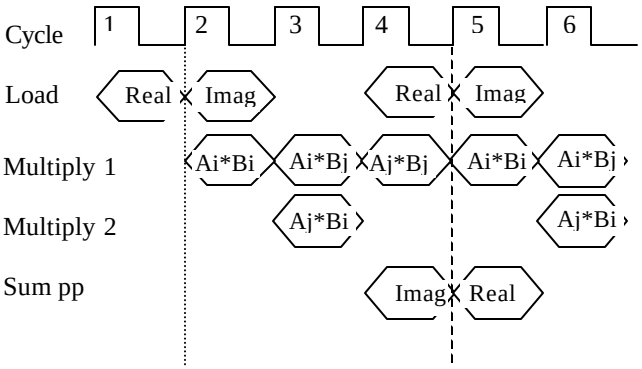

Figure5: A 3 Cycle Complex Multiply Operation using HCM-MAC

Some algorithms, such as FIR and DCT, that do not require the complex processing power of a HCM, map better onto a multiply accumulate architecture. It is often useful in a general purpose Dual MAC structure to be able to support application specific variants such as the HCM. The HCM based Execution unit architecture can be extended with minor modifications into a generic Dual MAC structure that includes two accumulators with multiplexed inputs that allow either one half-complex multiply with three clocks (HCM function) or two multiply

accumulates in parallel. Figure 6 shows the architecture for this general purpose Dual MAC (Dual ALU) Execution Unit. By selection of mux input select into the ALU (Add/Sub) blocks, either HCM or Dual MAC dataflow is supported.

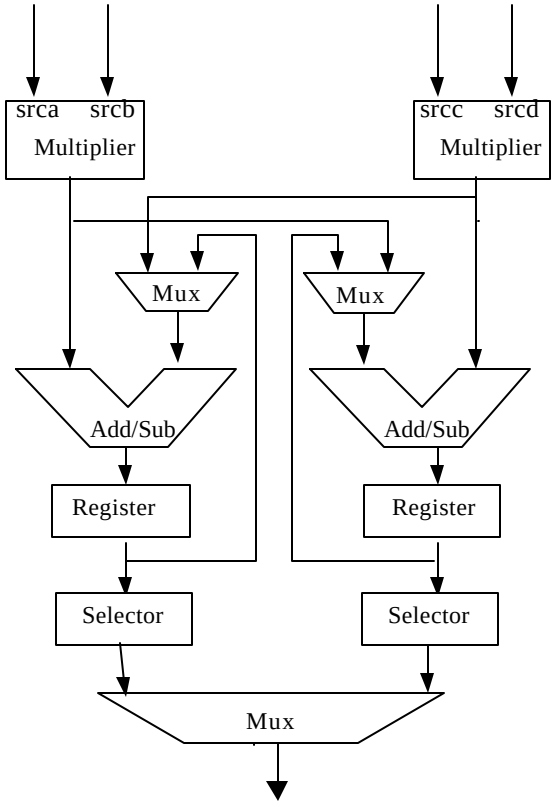

Figure 6: A Dual Multiply (DM-MAC) Execution Unit

	Half Complex Multiplier	Dual MAC/ (Dual ALU)
16 by 16 Multiplier	12000	12000
16 by 16 Multiplier	12000	12000
Adder / Subtractor	2000	
Selector	2000	
Mux(48)		500
Mux(48)		500
Adder / Sub Reg 1		2500
Adder / Sub Reg 2		2500
Selector		2500
Selector		2500
Mux(16)		200
Total	28000	35200

Table 1: Transistor Count for HCM vs. Dual MAC Execution Unit Architectures.

The typical silicon cost of HCM versus Dual MAC blocks is seen in Table 1, which compares approximate numbers of transistors for each MAC Execution unit. For 26 percent transistors increase compared to the HCM case, the MAC execution unit supports the generic flexibility of a Dual MAC architecture as well as a Half Complex multiplier.

4. A Quad MAC Complex Multiply Execution Unit

Dual MAC architectures have seen increased application in DSP architectures including the TI C6x and DSP Group Palm and Oak cores. General purpose DSP architectures supporting more than 2 MACs have only recently been proposed, notably in Lucent/Motorola Starcore and ADI TigerShark DSPs. A 4 MAC Execution Unit presents obvious advantages in addressing the complex multiply application previously discussed. A 4 MAC instance consisting of two HCM execution units plus four simple adders can allow FFT operation for a one clock Radix-2 butterfly with four clock latency, given sufficient memory bandwidth to prevent data starvation. This Execution Unit, integrated with 3 blocks of RAM as dedicated memory resources, is shown in Figure 7, as a typical example of a datapath IP product that addresses complex arithmetic oriented functions such as FFT, modulation, and demodulation.

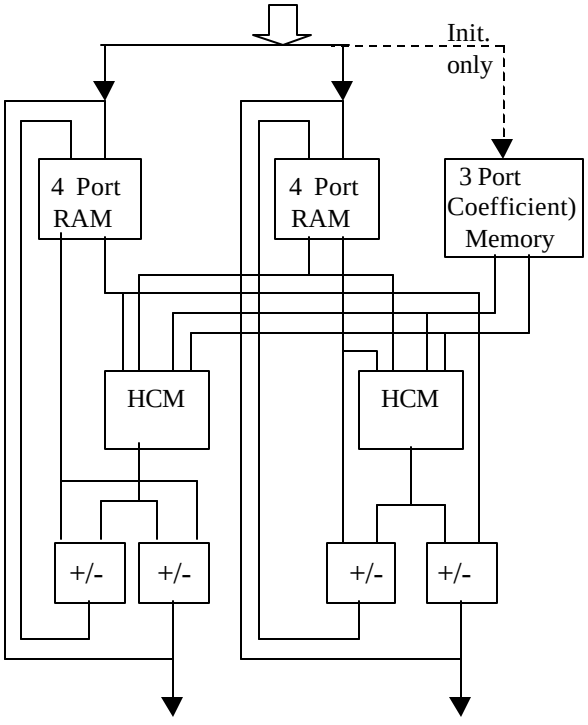

Figure7: A 4 MAC Single Cycle Complex Multiply Architecture using Dual HCM-MACs

In this dual HCM architecture, the data is read into the two 4-port memories sequentially. For example, the data for a

256 point complex transform would take 512 clocks to load the data and 512 clocks to store the results back to main memory via the I/O block. This architecture performs a Radix-2 butterfly per clock for a total of 1024 clocks of processing time. If the I/O cannot be overlapped, the total 256 point FFT time is 2048 clocks.

5. Summary

With the increasing availability of "cheap silicon" that the move to deep submicron technology is making available, the ability to customize high performance blocks such as MACs with enhanced supporting resources such as accumulators with ALU levels of functionality and dedicated memory becomes feasible. Infinite Technology's experience in this area indicates that, by trading off a measure of increased gate count and complexity for increased dataflow functionality and flexibility in memory support, performance in MAC intensive applications (a complex multiply algorithm being a typical example) can be increased significantly. MAC architectures based on the innovations discussed in this paper have been implemented as part of Infinite Technology's RAD (Reconfigurable Arithmetic Datapath) core customizable DSP architecture [2]. Since MAC operation is often the pacing element in the application specific DSP performance, understanding and evaluation of these tradeoffs can have a significant impact in the architectural selections and success of DSP SoC designs.

References

[1] S.Smith, R. Morgan, J. Rayne "ASIC Techniques for High Performance Digital Signal Processing" Ann. Telecommunications, v.46 #1-2, 1991 p 40-48

[2] "A High Performance Reconfigurable Signal Processor with Distributed IW Architecture" N. Stollon, G. Landers, T. Lahutsky, Proceedings of DesignCon99/IPWorld Forum, Feb. 1999 Additional info at ITC website http://www.itc-usa.com

Acknowledgements: Infinite Technology recognizes the contributions of Ralph Payne, employee-emeritus, whose insights and analysis, especially with regard to complex multiply and related applications, were key to the development of the multi-MAC architectures discussed in this paper. Infinite Technology also gratefully acknowledges the financial support of the Air Force SBIR program, administered by the Wright Patterson AFB Electronics Directorate, which provided partial funding for work that led to this paper.

Dedication: This paper is dedicated to the memory of Arthur Stollon, who passed away during the time of its creation. Among countless other things, thanks for all the years of proofreading.

Author Biographies

Neal Stollon is Director for Reconfigurable System-on-a-Chip efforts and a Senior Member of the Technical Staff at Infinite Technology Corp. Dr. Stollon has previously worked in diverse areas of digital design, design automation, and program management at Texas Instruments and DSC Communications. Dr. Stollon holds a Ph.D. in EE from Southern Methodist University and is a licensed P.E. (Texas). He has published numerous papers and holds several patents in areas of VLSI design and architecture and design automation.

Christopher Connell is a Senior Research Engineer at Infinite Technology Corp, where he is responsible for MAC architecture development and analysis. He has over 5 years experience in semi-custom and full-custom IC design, with an emphasis on HDL based design methodologies. Mr. Connell has an MS degree in Electrical Engineering from the University of Texas at Dallas (UTD), and is currently a Ph.D. candidate in Electrical Engineering at the University of Texas at Dallas in his third year of research.

George Landers is the VP of RAD technology and is the lead architecture design manager for Infinite Technology's patented RADcore technology. He has 35 years experience in the semiconductor industry with emphasis on product development and technical marketing of memory, reconfigurable logic and signal processing products. He was previously Manager of Strategic Marketing for PLD products and AMD and is a co-inventor of AMD's MACH product line. Mr. Landers holds 5 patents in the fields of programmable products and reconfigurable datapath products. Mr. Landers has a BSEE from the University of California at Los Angeles.