A Programmable Data-Path for MPEG-4 and Natural Hybrid Video Coding

Aamir A. Farooqui[1], Vojin G. Oklobdzija[2]

[1]Synopsys Inc.
Synopsys Module Compiler Group
700 Middlefield Road,
Mountain View CA 94043-4033
USA
(650) 584-5689
aamirf@synopsys.com

[2]ACSEL
Electrical and Computer Engineering Department
University of California
Davis, CA 95616
USA
(510) 486-8171
vojin@ece.ucdavis.edu

Abstract

A programmable data-path that supports MPEG standards Synthetic & Natural Hybrid video Coding (SNHC) is presented. It can support a maximum of 16 parallel SIMD integer operations and 2 parallel SIMD floating-point operations. Two new instructions were added in order to increase the execution of 3D graphics and SNHC as well as to speed up IDCT, FFT, and other media signal processing algorithms. These operations are implemented by re-using the hardware without significant increase in area and delay. The datapath has been modeled in Verilog using 0.25u CMOS library and synthesized using Synopsys. All operations are single-cycle running at 200 MHz.

1. Introduction

Low cost, low power, consumer electronics embedded media processors need a simple, and area efficient datapath that can support MPEG standards and Synthetic & Natural Hybrid video Coding (SNHC, or 3D graphics support) [1] [2] [3] [4] [5]. The goal of this research is to develop a reconfigurable datapath that can support integer as well as floating-point (FP\\0 operations for low cost, consumer electronics media processors.

This paper presents the design of a datapath organization and special SIMD instructions to support MPEG-1, 2, 4, SNHC and 3D graphics. The datapath is reconfigurable, scalable, area efficient, and offers programmability for interactive use. The reconfigurability and area reduction is achieved by the efficient re-use of same hardware instead of using redundant hardware as done in [6] [7]. The datapath can assist in the efficient and flexible coding and representation of both natural (pixel-based) as well as synthetic (computer generated) data (SNHC) by integrating FP operation support with integer operations. In order to support multiple data types with variable word lengths and formats, special techniques are developed to reconfigure the multipliers, adders, and to perform condition resolution.

The remainder of the paper is organized as follows. In Section 2 the overview of the proposed datapath is presented. In Section 3 integer instruction execution is described. Finally, results and comparisons are presented in Section 4.

2. Overview of the Datapath

In this Section, we present the datapath organization. The datapath supports operations on 128- and 64-bit signed unsigned integers and single precision floating-point numbers. Fig. 1 shows the block diagram of the datapath. This is a two-stage, pipelined datapath. The first pipeline stage contains four 16x32 partitioned multipliers (blocks 1, 2, 3, and 4), two 64-bit partitioned comparators, and three 12-bit adders for FP exponent calculation and 4x4 Sum of Absolute difference (SAD) instruction. The second pipeline stage contains two 64-bit partitioned shifters, adders, and circuitry for normalization and rounding.

The datapath instructions are divided into the following groups: 1) Add/Sub, Average, Shift, Logic and Permute 2) Multiply, Sum, Sum of Absolute difference, Pack and UNPACK, 3) Compare, Min, Max, Absolute value, and Absolute difference. All the instructions are SIMD type. All instructions have a latency and throughput of single cycle, except "Multiply, FP Multiply, FP Add, and SAD" that has a latency of two cycles and single cycle throughput. Following, we explain the implementation of some of these instructions using the proposed datapath.

3. Instruction Execution

In this section we present the instruction execution on the proposed datapath. Fig. 2 shows the macro-level implementation of the two cycle datapath, containing four 16x32 multipliers, 3x2 compressors, and two 64-bit adder-subtractors.

0-7803-6514-3/00/$10.00©2000 IEEE

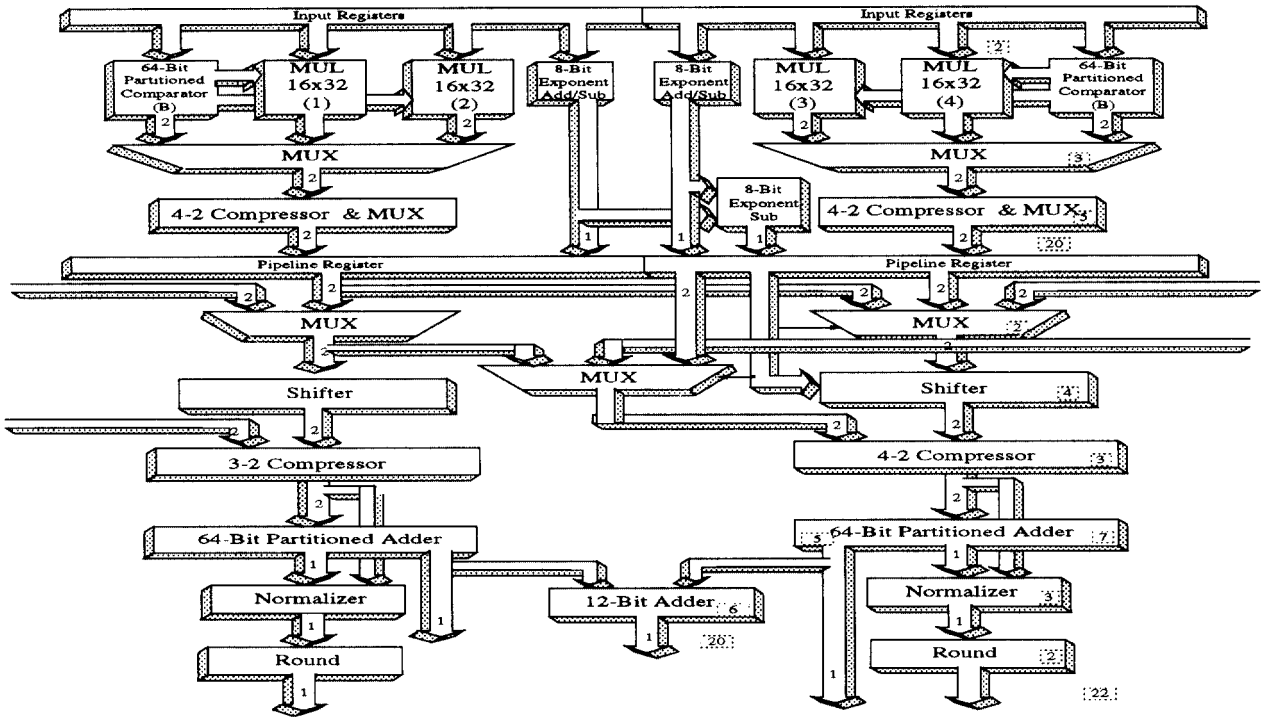

Fig. 1. Block diagram of the proposed Datapath.

3.1. Add/Sub Instructions

The Add/Sub instructions are executed in the second pipeline stage of the datapath using the 64-bit partitioned adders. The partition of the 64-bit adders is performed according to the partition control signals as shown in Table-2. Since we require two 64-bit adders for the multipliers, therefore by re-using these adders we can support two 64-bit operations in parallel. Table-1 shows the operation of these instructions on input operands A and B, the result is produced in C and D. The A, B, C, and D can be a byte, half-word, word, or double word packed as 8-bytes, 4-half-words, 2-words, and a double-word respectively; in a 64-bit register Rx, where x = 0...n.

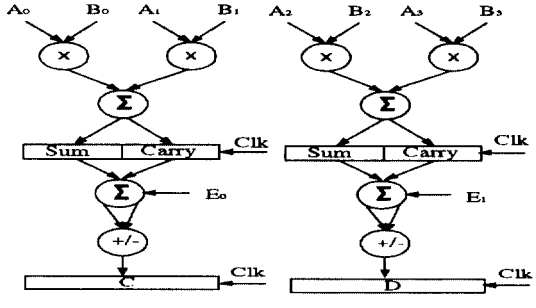

Fig. 2. Macro-level implementation of the datapath.

The ADD and SUB operations are performed as normal Addition and Subtraction (with and without saturation). The parallel average of two numbers is a very common and useful function in image and video processing (see Fig. 3-a). Only few high performance ISAs [15] support this operation. This combined operation involves an addition and a right shift of one bit (divide by two). In the proposed datapath, AVG2 operation is implemented using the 64-bit partitioned adder; performing normal addition and finally shifting the result one bit right. The datapath shifts in the carry out bit as the most significant bit of the result, so it has the added advantage that no overflow can occur.

Table-1. Instructions supported by the ALU (A, B, are the inputs, and C, D are the outputs).

Instruction	Operation	Result
ADD	A + B	C
ADS(U)	A + B	C or SAT(max)
ADS(S)	A + B	C or SAT(min) or SAT(max)
SUB	A-B	C
SUBS(U)	A-B	C
SUBS(S)	A-B	C or SAT(min) or SAT(max)
AVG2(U)	A+B/2	C/2
AVG2(S)	A+B/2	C/2

Table-2. Partition of the ALU, using Partition control signals.

Part1Part0	ALU partition
00	Byte
01	Half_word
10	Word
11	Double_word

3.2. Multiply Instructions

The multiply instructions and their macro operations are shown in Table-3. These instructions are executed using 32x16 multiplier blocks of the datapath. The partition of the

multipliers is performed according to the partition control signals as shown in Table-2 (11 is used for 24-bit operation). The multiplier hardware is configured for signed and unsigned operation using the sign control bit. When 'Sign' bit is '1' signed multiplication is performed and when it is '0' unsigned multiplication is performed. This group of instructions requires two-cycle latency with a single-cycle throughput. Following we explain the execution of multiply and sum of products instructions.

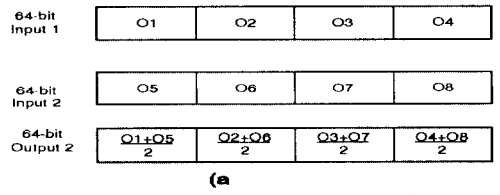

Fig. 3. Parallel averaging operation.

Multiply. In multiply operation, the two input operands are multiplied in the first cycle producing Sum and carry vectors, and in the second cycle Sum and carry vectors are finally added using the 64-bit partitioned adder to produce the final result as shown in Fig. 4. Since the result of multiplication is twice the width of the input operands, the result of simple multiplication is produced in C and D (C contains the lower half of the result and D contains the upper half of the multiplication). The only difference in MUL (U) and MUL(S), is that MUL(U) is the unsigned multiplication, while MUL(S) is the signed multiplication.

Table-3. Multiply operations supported by the proposed datapath.

Description	Inst.	Operation
Mult. with full resol. (S/U)	MUL	C = A*B
Mult. with accumulate (S/U)	MULA	C = A*B + C'
Mult. with deduct (S / U)	MULD	C = A*B - C'
Multiply upper half result with round. (S / U)	MULH	C = Ah*Bh
Floating point Multiply	FP MUL	(M0 x M1 , E0 + E1)
3D-Floating point sum of two products	3D-FSMUL	(M3 x M2 , E3 + E2) + (M0 x M1 , E0 + E1)
Sum of 2 Integer prod. (S/ U)	SUMP	A1*B1 + A0*B0
Sum of 2 products with accumulate (S / U)	SUMPA	$A_1*B_1 + A_0*B_0 + C'$
Sum of 2 products with deduct (S / U)	SUMPD	$A_1*B_1 + A_0*B_0 - C'$
Sum of 4 Integer prod. (S/ U)	SUM4P	$A_3*B_3+A_2*B_2+A_1*B_1+ A_0*B_0$
Sum of Absolute difference	SAD	$\Sigma \mid A - B \mid$
Sum of eight operands	SUM8	ΣA
Pack (S/ U) numbers	PK	
Unpack (S/ U) numbers	UPK	

SUM4P. Sum of two products is supported by most of the high performance media processors, but at present no processor exists that supports Sum of Four products. This operation performs the addition of four products and it is equivalent to four multiply and three additions as shown below:

$$C = A_0xB_0 + A_1xB_1 + A_2xB_2 + A_3xB_3$$

Matrix multiplication is heavily used in IDCT, FFT, and other media signal processing algorithms. We have found that this operation (SUM4P) dramatically increases (2 to 1) the execution of matrix multiplication by performing seven operations in a single cycle. Moreover, by using this operation, high precision processing is done because the rounding error is reduced as compared to two sum of products operation, in which two rounded results are added to get the addition of four products. While in this case full precision addition of the four products is performed and then rounding is performed.

The proposed datapath performs the addition of four products (SUM4P instruction) in two cycles with single cycle throughput. This operation is supported only for half-word operands. The instruction execution at macro level is shown in Fig. 5. This instruction requires four multiplications A_3xB_3, A_2xB_2, A_1xB_1, and A_0xB_0 these multiplications are performed in the first cycle using the MUL 32x16 blocks. The summation of two products (A_3xB_3 + A_2xB_2, and A_1xB_1+ A_0xB_0) is performed in the first cycle (Fig. 5) using the 4x2 compressor after the multipliers. Then, the summation of four 16x16 products is performed in the second cycle using the second 4x2 compressor. The sum and carry vectors produced by the summations are finally added using the 64-bit partitioned adder to produce the addition (A_0xB_0 + A_1xB_1 + A_2xB_2 + A_3xB_3) as shown in Fig. 5. In order to produce the result of the same bit width as the input the result of the addition is rounded and only the upper half of the result is stored in the output. As one can see from Fig. 5, Sum of four products operation requires the same datapath and components as required in Sum of two products for word operands, hence this instruction is implemented without any extra hardware cost and delay.

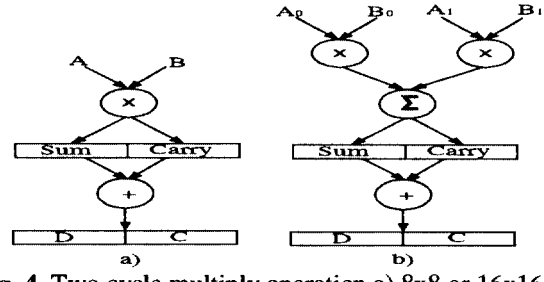

Fig. 4. Two cycle multiply operation a) 8x8 or 16x16, and b) 24x24 or 32x32.

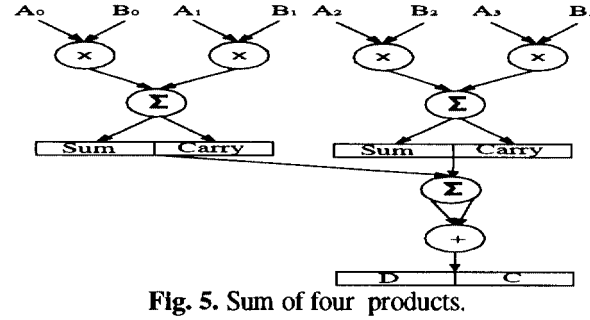

Fig. 5. Sum of four products.

4. Performance Estimation

The datapath performance is estimated for IDCT, Image reconstruction, and 3D Geometric Transformations using hand written assembly code. In this performance analysis it is assumed that the data is already available in the registers (loaded by the main processor or load store unit). Cache hit rate is assumed to be 100%, and instruction issue rate is one instruction/cycle. Add (ADD), Subtract (SUB), Average (AVG), Average of four operands (AVG4) and Permute (PERM) instructions operate on 128-bit data, while multiply related instructions operate on 64-bit data, Butterfly instruction perform two operations on two 64-bit operands and produce two 64-bit results.

The datapath requires, a total of 192 cycles to perform an 8x8 2D IDCT using matrix-vector implementation, while [5], [8], [9], and [4] requires 512, 448, 560, 500 respectively. This gives a performance gain of 2.33 over the current existing architectures. The main contribution to this performance gain is the new Sum of Four Products (SUM4P) instruction. Similarly, a total of (92 + 16 =) 108 cycles are required to perform 8x8 2D IDCT using Lee's Algorithm [10]. While [11] [11 optimized], [12], and [13] require 704, 504, 450, and 520 instructions respectively, giving it a performance gain of more than four times over existing architectures. The main contributions to this performance gain are the Butterfly, and 128-bit Add-Sub instructions, which constitute nearly 40% of Fast IDCT computation.

The datapath requires only 9 instructions/cycles to implement 3-D geometric transformation, while [5], [14], and [15] require 16, 16, and 13 instructions respectively. This is more than a 1.4 times increase in performance. This gain is due to the new FP Sum of Products (3D-FSMPY) instruction, which can perform two FP multiplications in parallel, thereby giving a performance gain of nearly 1.44 over the current existing architectures.

5. Conclusions

This paper presents an overview of a programmable datapath to support MPEG-1, MPEG-2, and MPEG-4 standard Synthetic & Natural Hybrid video Coding (SNHC). The datapath can support a maximum of 16 parallel SIMD integer operations and 2 parallel SIMD floating-point operations. The datapath is reconfigurable, scalable, and area efficient and offers programmability for interactive use. In this datapath, the reconfigurability is achieved by the efficient re-use of same hardware. The datapath can assist in the efficient and flexible coding and representation of both natural (pixel-based) as well as synthetic (computer generated) data (SNHC) by integrating floating-point operation support with integer operations. New instructions have been developed to speed up the execution of Matrix Multiplication, Discrete Cosine Transform (DCT), Inverse Discrete Cosine Transform (IDCT), Fast Fourier Transform, 3D-Graphics and Synthetic & Natural Hybrid Video Coding (SNHC). The datapath organization and development of algorithms, combined with circuit techniques yield performance improvement in media signal processing.

References:

[1] Nadehara, K., Kuroda, I., Daito, M., and Nakayama, T., "Low-power multimedia RISC", IEEE Micro Mag., vol. 15, pp. 20-29, Dec. 1995.

[2] Makino, H.; Suzuki, H.; Morinaka, H.; Nakase, Y.; Mashiko, K.,"A 286 MHz 64-bit floating point multiplier with enhanced CG operation", Symposium on VLSI Circuits, Digest of Technical Papers., 1995 , Page(s): 15 -16.

[3] Suzuki, K.; et. Al., "A 2000-MOPS embedded RISC processor with a Rambus DRAM controller", IEEE Journal of Solid-State Circuits, Volume: 34 7, July 1999, Page(s): 1010 –1021.

[4] Mohri, A.; et. al., , "A real-time digital VCR encode/decode and MPEG-2 decode LSI implemented on a dual-issue RISC processor", IEEE Journal of Solid-State Circuits, July 1999 , Page(s): 992 –1000 Volume: 34 7.

[5] Fujishima, H.; Takemoto, Y.; Onoye, T.; Shirakawa, I., "An architecture of a matrix-vector multiplier dedicated to video decoding and three-dimensional computer graphics", IEEE Transactions on CAS for Video Technology, Volume: 9 2, March 1999, Page(s): 306 -314.

[6] Schmookler, M.S., et. al., "A low-power, high-speed implementation of a PowerPC microprocessor vector extension", Proceedings 14th IEEE Symp. on Comp. Arith., 1999 , Page(s): 12 –19.

[7] Suzuki, K.; et. Al., "A 2000-MOPS embedded RISC processor with a Rambus DRAM controller", IEEE Journal of Solid-State Circuits, Volume: 34 7 , July 1999 , Page(s): 1010 –1021.

[8] Bum-Sik, K., Yun-Ho, C., Lee-Sup, K., "IRAM Design for Multimedia Applications", ISCA 1997, Page(s) 1-9.

[9] M. Yoshida, H. Ohtomo, and I. Kuroda, "A new generation 16-bit general purpose programmable DSP and its video rate application", in Proc. IEEE VLSI Signal Processing VI, Oct. 1993, pp. 93–101.

[10] C. Loefer, A. Ligtenberg, and G. S. Moschytz, "Practical fast 1-D DCT algorithms with 11 multiplications", in Proc. Int. Conf. Acoustics, Speech, and Signal Processing 1989 (ICASSP'89), pp. 988–991.

[11] Kuroda, I., "Processor architecture driven algorithm optimization for fast 2D-DCT", IEEE Signal Processing Society [Workshop on] VLSI Signal Processing, VIII, 1995, Page(s): 481 – 490.

[12] Holmann, E.; Yamada, A.; Yoshida, T.; Uramoto, S., "Real-time MPEG-2 software decoding with a dual-issue RISC processor", [Workshop on] VLSI Signal Processing, IX, 1996, Page(s): 105 –114.

[13] Peleg, A. and Weiser, U.,"MMX technology extension to the Intel architecture", IEEE Micro Mag., vol. 16, pp. 42-50, Aug. 1996.

[14] Radhika Thekkath, et. al.; "An Architecture Extension for Efficient Geometry Processing", Hot Chips 1999.

[15] Oberman, S., et. al.; "AMD 3Dnow! Technology: Architecture and Implementations", IEEE Micro, Page(s): 37-48, April 1999.