A 22-BIT FLOATING POINT REGISTERED ARITHMETIC LOGIC UNIT

Dr. John Eldon
Staff Engineer

TRW LSI Products
P. O. Box 2472
La Jolla, CA 92038

ABSTRACT

The LSI Products Division of TRW is currently developing a third chip for its growing family of 22-bit floating point arithmetic devices. Joining the adder and the multiplier later this year will be the registered arithmetic logic unit (RALU), built in TRW's dual-metal one-micron bipolar "Omicron-B" process. Operating at a guaranteed (military temperature and supply voltage ranges) speed of 6 MHz, this device will be able to store, retrieve, add, subtract, and normalize 22-bit floating point numbers, convert between 22-bit floating point and 16-bit fixed point formats, and add, subtract, and perform logical operations on 16-bit fixed point numbers. With its built-in shifters and controls, it can also perform a fixed point multiplication or division or a floating point division in 16 clock cycles.

The architecture of the RALU is very similar to that of the widely used 2901 four-bit microprocessor slice. The bus widths have been widened from 4 to 22 bits and the instruction set has been expanded to encompass the eight standard 2901 functions (for fixed point) and eight additional floating point and fixed-float conversion operations. The 2901's internal dual port RAM has been retained and widened for a 22-bit word size.

1. INTRODUCTION

Real-time digital signal processing (DSP) typically comprises numerous high-speed multiplications and additions of binary numbers. Quantization and arithmetic hardware limitations have traditionally constrained most of these operations to fixed-point arithmetic. However, fixed-point arithmetic has at least two significant deficiencies: 1) as numbers are accumulated, the probability of overflow steadily increases; and 2) the available relative precision depends strongly on the value of each number. Ameliorating either or both problems is costly. For example, one can eliminate overflow by increasing the widths of all parts of the data path, including buses, RAMS, and arithmetic elements. In addition to the obvious material cost penalties, a wider data path usually implies slower arithmetic element throughput.

Other systems avoid overflow by employing block-shifted fixed point or floating point arithmetic. A popular format for block fixed point is 16-bit two's complement, which permits a relative precision of up to 2^{-16}. The most prevalent floating point data word size has been 32 bits, typically with a 24 to 25-bit significand and an 8-bit exponent. Although 32-bit floating point requires twice the bus bandwidth of 16-bit fixed point arithmetic, it virtually eliminates both overflow and truncation errors.

For signal processing applications requiring high speed and moderate precision, TRW is introducing a line of monolithic floating point arithmetic units and multipliers in a 22-bit format (Figures 1 and 2). The significand is 16-bit two's complement, for compatibility with existing 16-bit fixed point host systems. The 6-bit two's complement exponent provides over 380 dB of dynamic range from 2^{-32} to 2^{+31}. The arithmetic units can convert between the 16-bit fixed-point and 22-bit floating point formats, allowing direct connection of a floating point processor board (with a signal path up to 22 bits wide) to the host system's 16-bit bus.

The first 22-bit floating point chip, the TDC1022 10 MHz arithmetic unit, is currently available. This part can add or subtract floating point numbers and convert between fixed and floating point formats. The TDC1042 multiplier chip, supporting both 16-bit fixed point and 22-bit floating point, will be introduced soon. The third chip, the TDC1033 registered arithmetic logic unit (RALU), scheduled for release later this year, is the subject of this paper.

2. RALU ARCHITECTURE

Figure 3 is a simplified block diagram of the RALU chip, showing the arithmetic logic unit, the register file, and the versatile bus structure connecting these elements. The chip can be regarded as a "floating-point 2901" with improved access to the register file and greatly enhanced arithmetic capabilities. The architecture and instruction set closely resemble those of the 2901, to simplify the adaptation of existing 2901 software to this new part.

The RALU's I/O structure permits the user to enter one operand and output one result on each clock cycle. During a single cycle, the chip can fetch one or two operands from memory, perform an arithmetic or logical operation, and store the result back in memory, while outputting the previous result or the contents of one memory register. Designed explicitly for 16-bit fixed point and 22-bit floating point operation, the RALU also supports 32-bit fixed-point addition and subtraction (in two clock cycles) and 22-bit floating point division (in a 15-cycle iterative subtraction algorithm).

2.1 INTERNAL BUS STRUCTURE AND SIGNAL ROUTING

As the block diagram illustrates, the bus structure is recursive, connecting the output of the RAM to the input of the ALU and the output of the ALU back into the input of the RAM and/or the ALU. The input multiplexing structure routes the contents of the input port to either the ALU or the RAM, a feature not available on the 2901. Thus, fresh operands can be loaded directly into chip memory, without passing through the ALU. The ALU source multiplexer selects one or both RAM outputs, the data input, or the recursive feedback line (via Register Q) as inputs to the ALU. The output multiplexer brings either the ALU's output or one of the RAM's read ports to the chip output port, so that the part can perform one operation while outputting any value stored in the RAM.

In use, common operations will include addition or subtraction with one operand from RAM and the other from the data input. Other possibilities are to normalize or denormalize the data input operand or one of the values in memory. The Q register path facilitates accumulation for filters and transforms.

2.2 THE ALU

Figure 4, a block diagram of the ALU portion of the RALU, shows the denormalizer, significand ALU, exponent comparator, renormalizer, and limiter. Various combinations of these sections are enabled to execute the chip's eight fixed-point and five floating point functions.

The denormalizing section contains a 16-bit rightward half-barrel shifter, through which one incoming significand is routed. This portion of the chip is disabled during fixed-point operation. Driving the denormalizing shifter is the exponent comparator, which subtracts one exponent from the other. The sign and magnitude of the difference tell which operand significand is to be downshifted, and how far.

The 16-bit ALU module operates on the significands as follows: A+B, A-B, B-A, A OR B, A AND B, A XOR B, A XNOR B, and NOT A AND B. The three arithmetic operations are used in both fixed and floating-point instructions; the five logical functions are strictly fixed-point.

The normalizer section contains a priority encoder and a 16-bit half-barrel shifter, as well as an exponent biaser to compensate for the normalization performed. In the fixed-point and denormalize modes, this portion of the ALU is disabled. The overflow and zero flags are generated here as well.

Finally, the user can program the limiter to replace each overflowing positive or negative quantity with the appropriate full-scale maximum value; similarly, underflowing values may be automatically replaced with "clean" floating-point zeroes. Gradual underflow is not part of the 22-bit format.

2.3 THE REGISTER FILE

The register file comprises sixteen 22-bit D-type registers, each with one master and two slave latches. The four-bit read/write address ("B") selects one of the 16 masters and one of the slaves. "A," the auxiliary read address, selects a second slave latch, which may be the same as the first. Since edge-triggered registers are used, write-in-place algorithms are readily executed: the user merely reads an operand from the RAM, operates on it in the ALU, and loads the result back into the same address in RAM. In the TDC1033, all registers are loaded on the rising edges of the system clock. Therefore, each instruction is executed during the interval between two successive rising clock edges.

3. INSTRUCTION SET

The RALU instruction set is a superset of those of the AM2901, AM2903, and TDC1022. As in the 2901, the instruction microcode can be conveniently divided into several fields: 1) ALU source code; 2) operand code; 3) destination code. These will be considered in turn, along with additional special codes peculiar to the RALU.

3.1 THE ALU SOURCE CODE

Figure 5 lists the eight states of the ALU source code, which drives the two multiplexers at the ALU's inputs. By manipulating the three code bits properly, one can select various pairs of operands from the contents of RAMA, RAMB, the Q (recursive) register, and the input port. Similarly, it is easy to zero out one ALU input for normalization, denormalization, and other single-operand functions. The zeroes supplied by the source multiplexer are "clean" floating-point zeroes, with zero significands and full-scale negative exponents, useful in both fixed and floating-point work.

3.2 ALU FUNCTION CODE

There are four ALU function control bits. One of these, the "fixed/float" mode control, selects between the eight 2901 fixed-point instructions and the five floating-point operations. As Figure 6 illustrates, the fixed-point mode provides the entire 2901 set of five logical and three arithmetic operations.

The floating-point instruction includes three pure (normalized) floating-point arithmetic operations: forward and reverse subtraction and addition. Floating-to-fixed point conversion (denormalization) is also provided, along with a special fixed-to-floating point conversion/addition function, in which the incoming exponents are zeroed, but the final result is a normalized floating-point value.

In the denormalization mode, the user enters one floating-point data value and a "seed" exponent. According to the difference between the exponents, the incoming significand is shifted rightward and output directly. If the data exponent exceeds the seed exponent, the significand overflow flag is set.

In the normalization/addition mode, the two incoming significands are added directly and assigned a zero exponent. When the sum is renormalized, this exponent is incremented or decremented accordingly. The result is a properly normalized floating-point number. This mode replaces the TDC1022's "normalize A" instruction, which can be emulated by feeding a zero into one side of the ALU and the desired operand into the other.

3.3 DESTINATION CODE

The destination code (Figure 7) not only selects the output data source (the RAM or ALU), but also determines the coupling between the RAM and the Q register. This facilitates double-precision fixed-point work in which the RAM generally holds the MSB's, while the Q holds the LSB's. With the internal carry linkages and instructions provided, double precision addition or subtrtaction merely requires operating first on the LSB field, applying any resulting carries, and then operating on the MSB field.

3.4 SPECIAL INSTRUCTIONS

To facilitate floating-point division, the RALU's fixed-point instructions contain an unusual feature. As one would expect, each selected fixed-point arithmetic operation is performed on the 16-bit significand field. Simultaneously, the difference between the two incoming exponents is calculated. This exponent difference may be ignored in most fixed-point work, but it becomes the quotient's exponent in floating-point division. Floating-point division is performed in a multicycle restorative subtraction sequence as follows:

1. On the first cycle, the divisor's exponent is subtracted from the dividend's, and the divisor's significand is subtracted from the dividend's, using the fixed-point subtraction instruction.
2. During successive cycles, the divisor's significand is shifted and subtracted from the intermediate remainder significands, while the original exponent difference is held in the Q register, using the DIV control.
3. At the end of the restorative subtraction sequence, the emerging quotient is sent through a normalization cycle, producing the desired floating-point result.

4. SUMMARY AND CONCLUSIONS

The RALU chip permits 6 MHz floating-point addition and subtraction, with the added benefit of internal scratch pad storage of intermediate results and coefficients. It is more versatile than the TDC1022, which merely adds, subtracts, and converts between formats. The RALU is recommended particularly for complex arithmetic, filters, and transforms, in which the on-chip storage can reduce the demands on the system bus, permitting enhanced throughput.

Unlike the TDC1022, the RALU provides a set of single-precision fixed-point instructions, plus simple sequences for double-precision fixed-point work. The RALU also provides special circuitry for floating-point division. In conjunction with the TDC1042 floating-point multiplier, the RALU can form the heart of a 6 MHz floating-point signal processor. The principal benefits of such a processor are its gargantuan dynamic range and simplified software. In particular, it can execute a complex FFT directly, without conditional (or unconditional) rescaling on each pass.

Figure 1. THE 22-BIT TWO's COMPLEMENT FLOATING POINT FORMAT

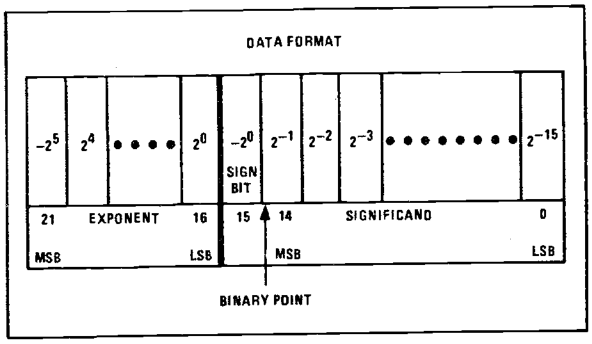

Figure 2. TRW 22-BIT FLOATING POINT FAMILY

NUMBER	NAME	FUNCTION	SPEED (Nsec)	AVAILABILITY
TDC1022	AU	+, -, NORM, DEN	100*, 200	NOW
TDC1042	FPM	X	100*, 160	SOON
TDC1033	RALU	+, -, LOGIC, NORM, DEN, -, DBL PREC	160	LATE '83

* With pipeline register enabled.

Figure 3. RALU FUNCTIONAL BLOCK DIAGRAM

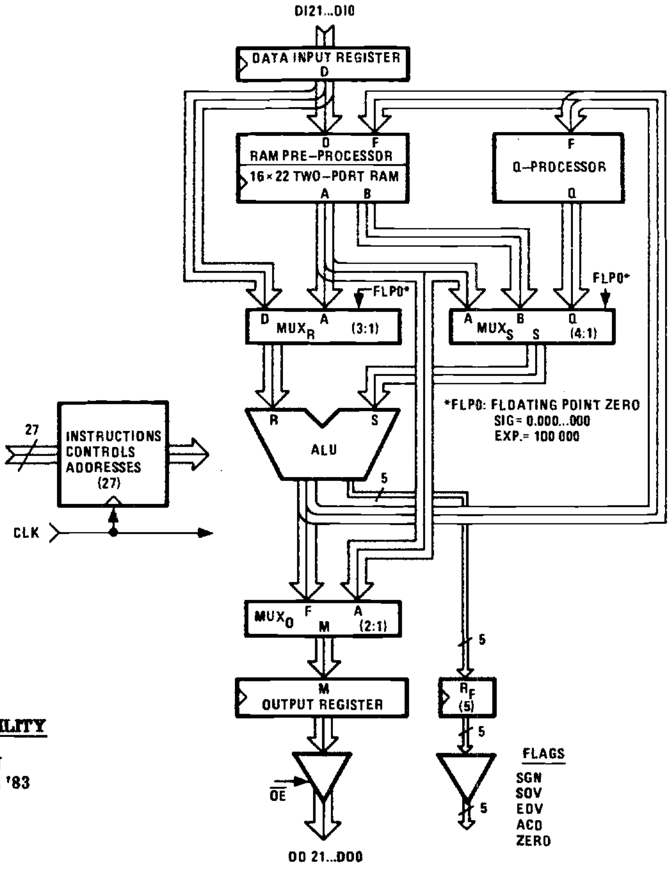

Figure 4. ALU SECTION BLOCK DIAGRAM

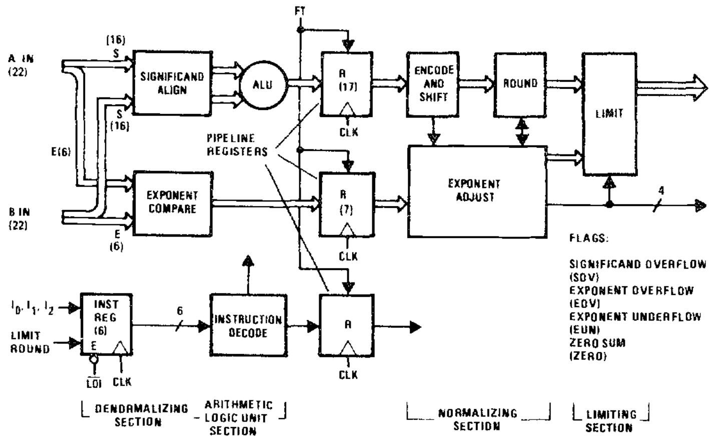

Figure 5. ALU SOURCE MULTIPLEXER

MNEMONIC	ALU INPUTS		MICRO CODE		
	R	S	I_2	I_1	I_0
AQ	A	Q	0	0	0
AB	A	B	0	0	1
ZQ	0	Q	0	1	0
ZB	0	B	0	1	1
ZA	0	a	1	0	0
DA	D	A	1	0	1
DQ	D	Q	1	1	0
DZ	D	0	1	1	1

NOTE: ALU Input of 0 is a floating-point zero i.e., SIGNIFICAND = 0.00..., EXPONENT = 10000.

Figure 6. ALU FUNCTION CONTROL

MNEMONIC	OPERATION	MICRO CODE				OUTPUT		NORMALIZATION	
		A_3	A_2	A_1	A_0	EXP	SIG	DENOR	RENOR
IADD	Add Fixed	0	0	0	0	R-S	R+S	Off	Off
ISUBR	Sub Fixed	0	0	0	1	R-S	S-R	Off	Off
ISUBS	Sub Fixed	0	0	1	0	R-S	R-S	Off	Off
OR	Logical OR	0	0	1	1	R-S	R or S	Off	Off
AND	Logical AND	0	1	0	0	R-S	R and S	Off	Off
NRS	R and S	0	1	0	1	R-S	R and S	Off	Off
XOR	Exclusive OR	0	1	1	0	R-S	R XOR S	Off	Off
XNOR	Exclusive NOR	0	1	1	1	R-S	RXNOR S	Off	Off
FADD	Add Floating	1	0	0	0	MAX(R,S)-	(R+S)'	EN	EN
FSUBR	Sub Floating	1	0	0	1	MAX(R,S)-	(S-R)'	EN	EN
FSUBS	Sub Floating	1	0	1	0	MAX(R,S)-	(R-S)'	EN	EN
ADDF	Add/Convert	1	0	1	1	0-	(R+S)'	Off	EN
DNOR	Denormalize R	1	1	1	0	MAX(R,S)	R"	EN	OFF

NOTES:

(R+S)'	Denotes a normalized (left-shifted as necessary) version of R+S.
MAX($,S)-	Denotes that the larger of the two operand exponents is selected, and then this exponent is decremented to accommodate the upward renormalization shift.
R"	Denotes a right-shifted version of the original operand's significand.
0-	Denotes that the output exponent value results exclusively from the action of the normalizer, i.e., the two incoming exponents are zeroed.
MAX(R,S)	Denotes the larger of the two exponents.

Figure 7. RALU DESTINATION CODE

DC MICRO CODE DC4-DC0 HEXIDECIMAL	MNEMONIC	RAM FUNCTION		Q-REG FUNCTION		OUTPUT	RAM SHIFTER INPUT		Q-REG SHIFTER INPUT		COMMENTS
		SHIFT	LOAD	SHIFT	LOAD		R_0	R_{15}	Q_0	Q_{15}	
00	QREG	X	NONE	NONE	F to Q	F	X	X	X	X	F to Q; F to Y
01	DRAMQ	X	D to B	NONE	F to Q	F	X	X	X	X	D to R; F to Q; F to Y
02	DRAMQA	X	D to B	NONE	F to Q	A	X	X	X	X	D to R; F to Q; A to Y
04	NOP	X	NONE	X	NONE	F	X	X	X	X	F to Y
08	RAMA	NONE	F to B	X	NONE	A	X	X	X	X	F to R; A to Y
09	DRAMA	X	D to B	X	NONE	A	X	X	X	X	D to R; A to Y
0C	RAM	NONE	F to B	X	NONE	F	X	X	X	X	F to R; F to Y
0D	DRAM	X	D to B	X	NONE	F	X	X	X	X	D to R; F to Y
10	RAMQ,RFZ	RIGHT	F/2 to B	RIGHT	Q/2 to Q	F	F_1	0	Q_1	0	Zero fill MSB of R, Q
11	RAMQ,RF1	RIGHT	F/2 to B	RIGHT	Q/2 to Q	F	F_1	1	Q_1	1	One fill MSB of R, Q
12	RAMQ,RCIR	RIGHT	F/2 to B	RIGHT	Q/2 to Q	F	F_1	F_0	Q_1	Q_0	Rotate R, Q right
13	RAMQ,RAR	RIGHT	F/2 to B	RIGHT	Q/2 to Q	F	F_1	F_{15}	Q_1	F_0	Arith. right shift RAM, Q
18	RAMQ,LFZ	LEFT	2F to B	LEFT	2Q to Q	F	0	F_{14}	0	Q_{14}	Zero fill LSB of R
19	RAMQ,LF1	LEFT	2F to B	LEFT	2Q to Q	F	1	F_{14}	1	Q_{14}	One fill LSB of R
1A	RAMQ,LCIR	LEFT	2F to B	LEFT	2Q to Q	F	F_{15}	F_{14}	Q_{15}	Q_{14}	Rotate R, Q left
1B	RAMQ,LAR	LEFT	2F to B	LEFT	2Q to Q	F	Q_{15}	F_{14}	0	Q_{14}	Arith. RAM, Q linked left

20.7