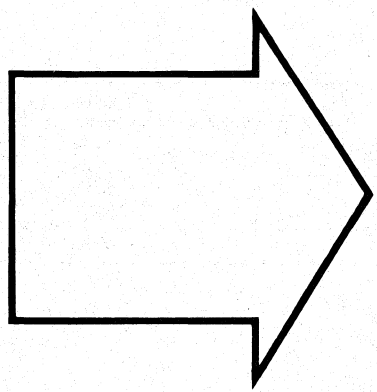


Table of Contents
MULTIPLIERS/DIVIDERS

Multiplier/Divider Selection Guide

Co-Processor Multiplier/Divider with Accumulator

	PART NUMBER	MAX MULTIPLICATION TIME/ MAX DIVISION TIME	PINS
8 Bits	SN74S508 SN54S508	0.8 μs/2.2 μs	24
16 Bits	SN74S516	1.5 μs/3.5 μs	24

Cray Multipliers

DESCRIPTION	PART NUMBER	MAX DELAY	PINS
8x8 Multiplier (latched)	SN74S557	60 ns (X_i, Y_i, to S_{15})	40
8x8 Multiplier (latched)	SN54S557	60 ns	40
8x8 Multiplier (latched)	SN74S558	60 ns	40
8x8 Multiplier (latched)	SN54S557	60 ns	40
16x16 Multipliers	SN74S556	90 ns	84

Five New Ways to Go Forth and Multiply

Chuck Hastings

Our Multiplier Population Explosion

Recently it has seemed as if every time you turned around Monolithic Memories was announcing *another* new multiplier. Want to catch your breath, and find out where each of these fits into the overall scheme of things? Read on.

Actually, there have been five new multipliers in all within the last three years, plus two which had previously been available for several years. In time order of introduction, these are:

Parts No.	Description[A]
57/67558	150-nsec 8x8 Flow-Through Cray Multiplier[B]
57/67558-1	125-nsec 8x8 Flow-Through Cray Multiplier[B]
54/74S508	8-Bit Bus-Oriented Sequential Multiplier/Divider
54/74S558	60-nsec 8x8 Flow-Through Cray Multiplier
54/74S557	60-nsec 8x8 Flow-Through Cray Multiplier with Transparent Output Latches
54/74S516	16-Bit Bus-Oriented Sequential Multiplier/Divider
54/74S556	90-nsec 16x16 Flow-Through Cray Multiplier with Transparent Input and Output Latches

NOTES: A. Times are worst-case times for commercial-temperature-range parts.
B. Obsolete. 54/74S558 replaces these in both new and existing designs.

You will notice that the above parts fall into two categories: flow-through Cray multipliers, and bus-oriented sequential multiplier/dividers. Although all of these parts get referred to rather casually as "multipliers," there are major differences between the two general types; see Table 1 below.

The Cray Multipliers

The essential idea of a Cray multiplier, as originally put together by Seymour Cray in the late 1950s with discrete logic at Control Data Corporation, is to wire up an array of full adders in the form of a binary-arithmetic-multiplication pencil-and-paper example.[3] That is, everywhere that there is a "1" or a "0" in a longhand binary-multiplication example, the Cray type of multiplication uses a full adder. One may visualize a Cray multiplier functionally as a "diamond," as follows:

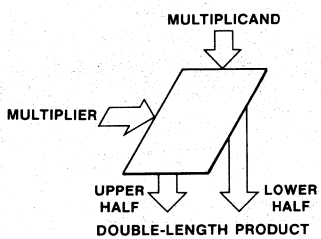


Figure 1. Pencil-and-Paper Analogy to Cray-Multiplier Operation

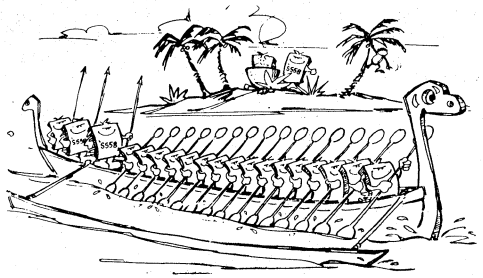
10

	Flow-Through Cray Multiplier	Bus-Oriented Sequential Multiplier/Divider
Role in System	*Building-block role* — as many as 34 parts used in one super-minicomputer (NORD-500 from Norsk Data[1]).	*Co-processor role* — one, or occasionally two, parts used in one microcomputer[2].
Internal Operation	Static arithmetic-logic network; multiplies without being clocked[3] using eight bits of the multiplier at a time.	State machine; requires clocking to operate; contains edge-triggered registers; sequenced by a state counter; multiplies using two bits of the multiplier at a time[4].
External Control	Controlled by several mode-control input signals.	Controlled by sequences of micro-opcodes which come from a microprocessor, a registered PAL, or some other sequential-control device.
Package	40-pin DIP ('S557/8); 84-pin LCC or 88-pin PGA ('S556)	24-pin DIP.
Operations Performed	Can only perform multiplication.	Can perform multiplication, division, and multiplication-with-accumulation.
Storage Capabilities	Either no storage capabilities ('558 types), or optional storage for the double-length product only ('557 type), or full product and input storage ('556 type).	Four full-length registers; capable of storing both input operands and the double-length product.
Second Sources	8x8, Multiple-sourced (AMD, Fairchild, Monolithic Memories).	Sole-sourced; only bipolar *dividers* on the market.
Where Used	Initial usage has been in high-end minicomputers, array processors, and signal processors.	Initial usage has been in industrial-control microcomputers, digital modems, military avionics, CRT graphic systems, video games, and cartographic analysis systems.
Future Prospects	Potential large market today since these parts are now low-cost and multiple-sourced, and should be used in *all* new minicomputer designs!	Potential huge world-wide market for enhancement of microprocessor, bit-slice processor, and microcomputer capabilities, and for small-scale signal processing!

Table 1. A comparison of the two types of Monolithic Memories Multipliers

Our 57/67558, introduced in the mid-1970s, was the original *single-chip* Cray multiplier. To achieve what was for that time very high performance for a Schottky-TTL-technology part, the internal design of the 57/67558 also exploited other speed-freak multiplication techniques such as Booth multiplication[4] and Wallace-Tree addition[5]. All of these techniques achieve increased speed through extensive parallelism, and can be used at the system level as well as within LSI components. Subsequently, process improvements made it possible to offer a faster final-test option, the 57/67550-1, which attained a sales-volume level essentially equal to that of the original part.

About five years ago, AMD paid us the sincere compliment of second-sourcing these parts with the 75-nsec 25S558. Three years ago, we returned the compliment with the 60-nsec 54/74S558. All of these '558 parts, and the 70-nsec 54/74F558 announced by Fairchild, are fully compatible drop-in equivalents except for the variations in logic delay.



"ALL OF THESE TECHNIQUES ACHIEVE INCREASED SPEED THROUGH EXTENSIVE PARELLELISM."

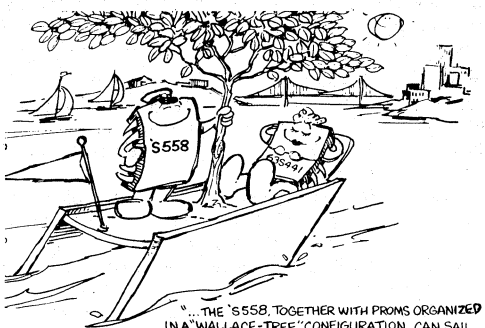
When AMD introduced the 25S558, they introduced along with it the 80-nsec 25S557, a "metal option" of the same basic design with "transparent" output latches to hold the double-length product. "Transparent" means that the latches go away when you don't want them there; a latch-control line like that of the 54/74S373 controls whether these output latches store information, or simply behave as output buffers. Anyway, when we introduced our 54/74S558, we followed it within a few weeks with the 60-nsec 54/74S557, which is a much faster drop-in replacement for AMD's part. And subsequently, Fairchild has announced a 70-nsec 54/74F557.

Because AMD's 'S557 has the output latches implemented in TTL technology *after* the ECL-to-TTL converters, whereas our 'S557 has them implemented in ECL technology *before* the conversion, the latches operate much faster in ours. Our 'S557 is typically only about a nanosecond slower than our 'S558, whereas the logic-delay difference between AMD's two parts is considerably greater. Consequently, our margin of superiority over AMD for the 'S557 is even greater than for the 'S558.

More recently, we introduced the 90-nsec 'S556, which is a 16x16 direct size-upgrade of the 'S557/8 architecture, with the addition of input latches. In a "pipelined" mode, an 'S556 can produce a new 32-bit product every 75 nsec.

'S557/8 Cray multipliers come in a 40-pin dual-inline package, either ceramic or plastic. Worst-case power-supply current is 280 mA. The 'S556 comes in your choice of an 84-pin LCC (*L*eadless Chip *C*arrier) or an 88-pin PGA (*P*in-*G*rid *A*rray) package. Worst-case power-supply current is 800 mA (900 mA over military temperature range). The data-bus outputs can sink up to 8 mA I_{OL}, for all of these multipliers.

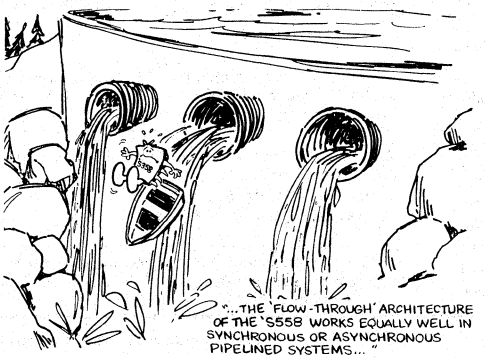
References 5 and 6 discuss technical approaches to using Cray multipliers in high-performance minicomputers. The 'S558, together with PROMs organized in a "Wallace-tree" configuration, can sail right along at the rate of four 56x56 multiplications every microsecond, on the basis of fixed-point arithmetic with no renormalization. (See table 7 on page 16 of reference 5; the multiplication time is 238 nsec for a "division step," which is a fixed-point multiplication, and 319 nsec for a floating-point multiplication where extra time is required for renormalization and correction of the exponent of the product.) 34 'S558s or 'S557s are required to perform this multiplication if the computer system architecture does not call for the computation of the least-significant half of the double-length product; 49 are required if it does.



"...THE 'S558, TOGETHER WITH PROMS ORGANIZED IN A 'WALLACE-TREE' CONFIGURATION, CAN SAIL RIGHT ALONG AT THE RATE OF FOUR 56 x 56 MULTIPLICATIONS EVERY MICROSECOND..."

The "local" architecture of the multiplier section of a digital system can take two rather different forms. A minicomputer[5], which executes an unpredictable mixture of arithmetic and logical instructions one after the other, typically needs to be able to get the complete multiplication over and done with before going on to the next program step—which is probably not another multiplication. An array processor or digital correlator, however, tends to do very regular iterative computations; and the performance of such a system can often be greatly increased by a technique called "pipelining," in which the arithmetic unit consists of stages with registers or latches in between each stage, and partial computational results move from one stage to the next on each clock.

The "flow-through" architecture of the 'S558 works equally well in synchronous or asynchronous pipelined systems, but registers or latches must be provided externally. The 'S557, however, is actually a *superset* of the 'S558, and the added internal-output-latch feature adapts it particularly well to pipelined systems. The 'S556 provides latches at *both* ends.



"...THE 'FLOW-THROUGH' ARCHITECTURE OF THE 'S558 WORKS EQUALLY WELL IN SYNCHRONOUS OR ASYNCHRONOUS PIPELINED SYSTEMS..."

Even a smaller-scale system can make effective use of these parts. To return to the case of 56x56 multiplication, which corresponds to the word-length needed for multiplying mantissas in several popular floating-point-number formats, an iterative clocked scheme using just seven 8x8 multipliers, some adders, and an accumulator register can form the entire 112-bit double-length product in just seven multiply/add cycles. A number of mid-range minicomputers today multiply in this manner. The multipliers are configured as suggested by the following block diagram:

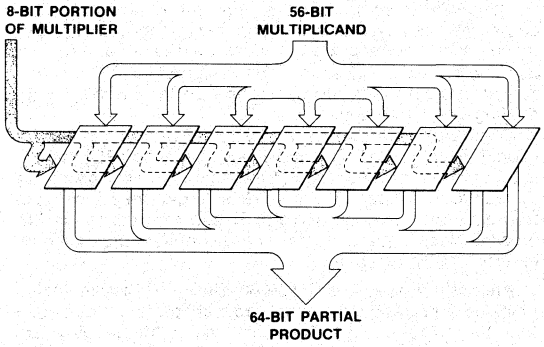


Figure 2. 8x56 Cray Multiplier In Diamond Representation

There is even an occasional 8-bit or 16-bit microprocessor-based system with a need for *very* fast multiplication, where 'S557/8s or 'S556s may get used as microprocessor peripherals[7,8]. Digital-video systems, in particular electronic games, with "vector graphic" capabilities are one example.

The world of 'S556/7/8 applications has turned out to include *all* sizes of minicomputers, digital video systems, and signal processors — FFT (Fast Fourier Transform) processors, voice recognition equipment, radar systems, digital correlators and filters, electronic seismographs, brain and body scanners, and so forth. And there are many unexpected off-beat applications, such as real-time data-rescaling circuits in instruments, altogether too numerous to list here. After all, an'S556 can multiply two 16-bit numbers together and output their entire 32-bit product in 90 nsec *worst case*...less time than it would take a speeding bullet to move the distance equal to the thickness of this piece of paper. How's that for Supermultiplier?

The Multiplier/Dividers

The Monolithic Memories 'S516 and 'S508 are state-of-the-art TTL-compatible intelligent peripherals for microprocessors, somewhere between arithmetic sequential circuits and specialized bipolar microprocessors. The 'S516 and 'S508 each can perform any of 28 different multiply and multiply-and-accumulate instructions, plus any of 13 different divide instructions, at bipolar speeds under the control of an internal state counter. (See Figure 2 of the 'S516 data sheet.) The state counter's sequence is in turn guided by 3-bit instruction codes which are external inputs to the 'S516/508. The 'S516 computes with 16-bit binary numbers, and the 'S508 computes with 8-bit binary numbers, as the part numbers none-too-subtly imply.

A 16-bit bi-directional data bus connects the 'S516 with the outside world for bringing in multipliers, multiplicands, dividends, and divisors; and returning products, quotients and remainders. It also has clock (CK) and run/wait ($\overline{GO}$) inputs, and an overflow indication (OVR) output.

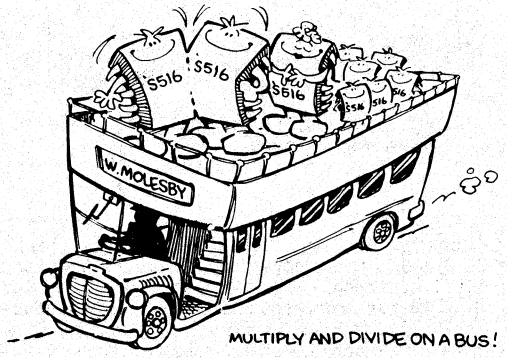
The 'S508 has all of the above inputs and outputs also, except that it has only an 8-bit bidirectional data bus. Since it comes in the same 24-pin package as the 'S516, it obviously has eight more pins available for other purposes. Four of these are used to bring out the internal-state-counter value; one each is used for a completion ($\overline{DONE}$) status output, an output-enable control (OE) input, and a master-reset ($\overline{MR}$) control input; and one is not used at all.

A simple, general interfacing scheme can be used to team a 'S516 with any of the currently popular 16-bit microprocessors,or an 'S508 with any 8-bit microprocessor. (See Figure 7 of the'S516 data sheet.) With a couple extra interface circuits, an'S516 can also be interfaced to an 8-bit microprocessor.Particularly if the system software is written in a highly-structured language such as PASCAL or FORTH, an'S516/508 can be retrofitted into an existing system with a large gain in performance and very little impact on either hardware or software — calls to the previous software-implemented one-step-at-a-time multiply and divide subroutines are simply rerouted to substitute a command from the microprocessor to the 'S516/508 to accept an operand and start its operation sequence.

The 'S516 and 'S508 are in fact two different "metal options" of one basic design; the 'S516 has twice as many data bits in each internal register. The 'S516 and 'S508 both have a worst-case clock rate of 6 MHz (commercial) or 5 MHz (military); the typical rate is 8 MHz. The simplest complete twos-complement 16x16 multiplication instruction can be performed in nine clock cycles by an 'S516, or in five by an 'S508, since 2-bits-at-a-time Booth multiplication is used;[4] thus, the worst-case time required by the 'S516 to multiply in this mode is 1.5 μsec for a commercial part, and for an 'S508 it is 833 nsec. On the same basis, 32/16 division can be done in 21 clock cycles, or 3.5 μsec worst-case, by an 'S516; and 16/8 division can be done in 13 clock cycles, or 2.2 μsec worst-case, by an 'S508.

An 'S516/508 can perform either positive or negative multiplication or multiply-accumulation, and many of the instructions provide for "chaining" of successive computations to eliminate extra operand transfers on the bus; these features further enhance the computational speed of the 'S516/508 in particular applications. Arithmetic can be either integer or fractional with respect to positioning of the results.

An 'S516 can powerfully enhance the capabilities of *any* present-day 16-bit or 8-bit microprocessor in a compute-bound application. In fact, it can be used in any digital system where there is a need to multiply and divide on a bus. An 'S508 can likewise enhance the capabilities of any 8-bit microprocessor.



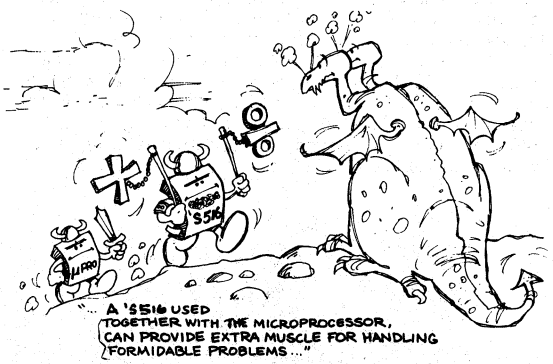
10

The 'S516 comes in an industry-standard 600-mil 24-pin dual-inline package, modified to include an integral aluminum heatsink which does not add appreciably to the package height. It requires only +5V and ground power connections, and draws a worst-case power-supply current of 450mA (commercial) or 500mA (military). Power consumption is greatest at cold temperatures, and decreases substantially as operating temperature increases. The 16 databus inputs require at most 0.25mA input current; the other inputs require at most 1mA. The 16 databus outputs can sink up to 8mA I_{OL}. The 'S508 also fits the above description, except that its worst-case power-supply current is 380mA (commercial) or 400mA (military), and it has only 8 databus inputs and outputs.

In describing applications of these parts, it is difficult to know where to start — they can be used in almost any design where a microprocessor can be used, and you know how many places that is today. So, perhaps a good starting point is to see what uses customers have thought up all by themselves. One customer even used *two* 'S516s in "pingpong" mode on a single 16-bit bus! So, rather than merely speculating as to what these parts *might* be good for, here's a list of what Monolithic Memories's customers have already *proven* they are good for:

- Real-time control of heavy machinery[9]
- Low-cost, high-performance digital modems
- CRT graphics, including video games
- Military avionics
- Cartographic analysis

As it happens, the above are 'S516 applications, except that digital modem designs have been done with both the 'S516 and the 'S508. Several of the 'S516 designs are already in production. In each of these applications, the microprocessor could have coped all right with the computational complexity, albeit at its own less-than-tremendous speed, but a 'S516 used together with the microprocessor can provide extra muscle for handling formidable problems.



Competition? Well, since there are no second sources for the 'S516, and no competitor at present has a similar fast part capable of performing division as well as multiplication, right now the 'S516 has no *direct* competition. Indirectly, there are some competing parts which perform *only* multiplication, and would have to perform division by Newton-Raphson iteration to be usable for any application where division is required. However, the 'S516 is (as far as we know) by far the lowest-priced *bipolar* 16-bit multiplier, and the other microprocessor peripheral chips which can perform division as well as multiplication are relatively-slow MOS devices. In one case, an 8-bit cascadable CMOS part requires a 50% reduction in clock rate to do 16-bit arithmetic. And considerable numerical-analysis and programming sophistication are required to implement Newton-Raphson division with *fixed-point* operands. (It's easier with floating-point operands.) In contrast, the 'S516/508 can be easily interfaced to almost any microprocessor using one or two PALs,® and can perform *either* multiplication or division on command?

The 'S516 is so much faster than the competing MOS chips that it can even take them on for *floating-point* computations (which some of them are designed to do) and *win* . A conference paper[10] describes the design of an 'S516-based S-100-bus card capable of beating an Intel 8087 2:1 on floating-point arithmetic.

Some competing parts, in particular the AMI 2811 and Nippon Electric μPD7720, include an on-board ROM which must be mask-programmed at the factory, which makes life difficult for small companies (or even larger ones) which are trying to get a microprocessor-based product to market quickly. Also, some competing parts require sequencing by external TTL jellybeans.

And, as for using AMD/TRW 64-pin 16x16 Cray multiplier chips as microprocessor peripherals, these cost much more than the 'S516, occupy about three times the circuit-board space, multiply faster, don't divide at all except by Newton-Raphson iteration, and also require one or two "overhead" microprocessor instructions to interface for a given arithmetic operation. From a *system* viewpoint, when this overhead time is reckoned with, these chips provide little actual gain in multiply performance over the 'S516 at lots of extra cost, and an actual loss in divide performance: the 'S516 is *much more cost-effective overall.*

'S516s potentially fit into many, many places in commercial, industrial, and military electronics, particularly into small-scale real-time systems. The part is fast enough to enhance the performance of a 16-bit Motorola 68000, Zilog Z8000, or Intel 8086, as well as that of *any* 8-bit microprocessor. It is also fast enough to considerably improve the multiplication and division performance of 16-bit 2901-based "bit-slice" bipolar microcomputers, which are often used as processors in desktop graphics CRT terminals.

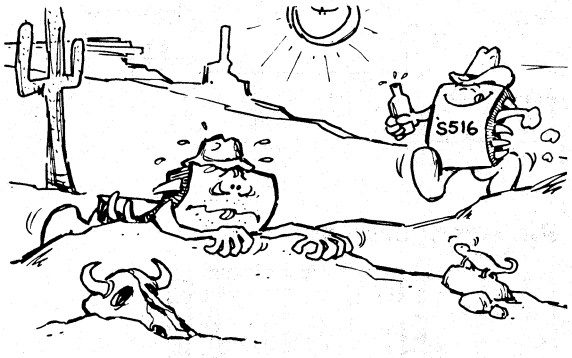
It is worth bringing the 'S516 to the attention of *any* designer who is developing:

- A personal computer or small business computer.
- A word processor, or a more grandiose "office automation system."
- A cruise missile, or any other "smart weapon."
- A digital modem.
- A small-scale speech-processing system. (These are *very* multiplication-intensive. We have one magazine article on the 'S516 in such an application![11])
- A smart instrument, which does data conversion.
- An industrial control system, particularly one which must do many coordinate transformations.
- An all-digital studio-quality high-fidelity system.
- A cost-reduced computerized medical scanning system.
- A multiprocessor system for scientific computations![12])

If an 'S516/508 is introduced into a system configured around an older microprocessor as a "co-processor" or

helpmate for the microprocessor, and the application is arithmetic-intensive, the end effect can be a major upgrading of performance at the system level.[2-7] Consequently, a major reason for designing these parts in is *microprocessor life-cycle enhancement*. In particular, many MOS microprocessors have single-length and double-length add and subtract instructions: but either they have no multiply or divide instructions at all, or else they perform their multiply and divide instructions so slowly as to jeopardize the ability of the entire system to handle its computing load in real time.

So picture, if you will, the entrepreneur or chief engineer of a firm making a successful microprocessor-based widget which has been on the market for a few months, which uses an older 8-bit microprocessor such as a 6800 or 8085 or Z80. Just when his/her sales are really taking off, here comes a new start-up competitor with a similar system, using a Motorola 68000, with added features and faster performance made possible by the 68000's 16-bit word length and multiply/divide capabilities. The 'S516 can, in this instance, serve as a "great equalizer"—it can be retrofitted into the older system as previously described, and provides even higher-speed multiplication and division than the 68000. (Enough so, actually, that there are designers using the 'S516 *with* the 68000.) Thus, the 'S516 can dramatically extend the life cycle of existing microcomputer systems based on microprocessors which either don't have multiplication and division instructions, or perform these operations relatively slowly.



"... THE 'S516 CAN DRAMATICALLY EXTEND THE LIFE CYCLE OF EXISTING MICROCOMPUTER SYSTEMS BASED ON MICROPROCESSORS WHICH EITHER DON'T HAVE MULTIPLICATION AND DIVISION INSTRUCTIONS, OR PERFORM THESE OPERATIONS RELATIVELY SLOWLY..."

'S508s are somewhat easier to control from a logic-design viewpoint than 'S516s, purely because they have more control inputs and outputs. However, the shorter 'S508 word length makes the part naturally fit into smaller-scale systems than those which might use an 'S516. Essentially, the 'S508 is optimized for small-scale systems.

Now that you know what these parts are, can't you think of at least half a dozen prime uses for them right in your own back yard?

References (all available from Monolithic Memories)

1. "Combinatorial Floating Point Processor as an Integral Part of the Computer," Tor Undheim, *Electro/80 Professional Program Session Record,* Session 14 reprint, paper 14/1.
2. "SN54/74S516 Co-Processor Supercharges 68000 arithmetic," Richard Wm. Blasco, Vincent Coli, Chuck Hastings and Suneel Rajpal, Monolithic Memories Application Note AN-114.
3. "How to Design Superspeed Cray Multipliers with 558s," Chuck Hastings, included within the SN54/74S557/8 data sheet.
4. "Doing Your Own Thing in High-Speed Digital Arithmetic," Chuck Hastings, Monolithic Memories Conference Proceedings reprint CP-102.
5. "Big, Fast, and Simple — Algorithms, Architecture, and Components for High-End Superminis," Ehud Gordon and Chuck Hastings, Monolithic Memories Application Note AN-111.
6. "Fast 64x64 Multiplication using 16x16 Flow-Through Multipliers and Wallace Trees," Marvin Fox, Chuck Hastings and Suneel Rajpal, Monolithic Memories Conference Proceedings reprint CP-111.
7. "An 8x8 Multiplier and 8-bit μp Perform 16x16 Bit Multiplication," Shai Mor, *EDN,* November 5, 1979. Monolithic Memories Article Reprint AR-109.
8. "Using a 16x16 Cray Multiplier as a 16-Bit Microprocessor Peripheral to Perform 32-Bit Multiplication and Division," Chuck Hastings, Monolithic Memories Conference Proceedings reprint CP-140
9. "The Design and Application of a High-Speed Multiply/Divide Board for the STD Bus," Michael Linse, Gary Oliver, Kirk Bailey, and Michael Alan Baxter, Monolithic Memories Application Note AN-115.
10. "Minimum Chip-Count Number Cruncher Uses Bipolar Co-Processor," C. Hastings, E. Gordon, and R. Blasco. Monolithic Memories Conference Proceedings reprint CP-109.
11. "Medium-speed Multipliers Trim Cost, Shrink Band-width in Speed Transmission," Shlomo Waser and Allen Peterson, *Electronic Design,* February 1, 1979; pages 58-65. Monolithic Memories Article Reprint AR-107.
12. "A Synchronous Multi-Microprocessor System for Implementing Digital Signal Processing Algorithms," T.P. Barnwell, III and C.J.M. Hodges, *Southcon/82 Professional Program Session Record,* Session 21 reprint, paper 21/4.

8x8 Multiplier/Divider
SN54/74S508

Features/Benefits

- **Co-processor for enhancing the arithmetic speed of all present 8-bit microprocessors**
- **Bus-oriented organization**
- **24-pin package**
- **8/8 or 16/8 division in less than 2.2 μsec**
- **8x8 multiplication in less than .8 μsec**
- **28 different multiplication instructions such as "fractional multiply and accumulate"**
- **13 different divide instructions**
- **Self-contained and microprogrammable**

Description

The SN54/74S508 ('S508) is a bus-organized 8x8 Multiplier/Divider. The device provides both multiplication and division of 2s-complement 8-bit numbers at high speed. There are 28 different multiply options, including: positive and negative multiply, positive and negative accumulation, multiplication by a constant, and both single-length and double-length addition in conjunction with multiplication. 13 different divide options allow single-length or double-length division, division of a previously-generated result, division by a constant, and continued division of a remainder or quotient.

The 'S508 is a time-sequenced device requiring a single clock. It loads operands from, and presents results to, a bidirectional 8-bit bus. Loading of the operands, reading of the results, and sequential control of the device is performed by a 3-bit instruction field.

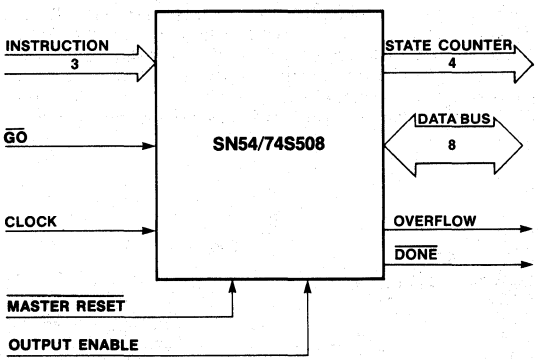
The 'S508 has the additional feature that operands and results can be either integers or fractions; when it deals with fractions, automatic scaling occurs. Results can be rounded if required, and an Overflow output indicates whenever a result is outside the normally-accepted number range.

For a simple multiplication of two operands and reading of the double-length result, the device takes five clock periods — one for initialization, and four for the actual multiplication. A typical clock period is 125 ns, which gives a multiplication time of 500 ns typical for 8x8 multiplication, plus 125 ns additionally for initialization, or 625 ns in all. More complex multiplications will take additional clock periods for loading the additional operands. A simple division operation requires 8 + 4 = 12 clock periods for a typical time of 1.5 μs (16 bits/8 bits), also plus 125 ns for initialization, or 1.625 μs in all.

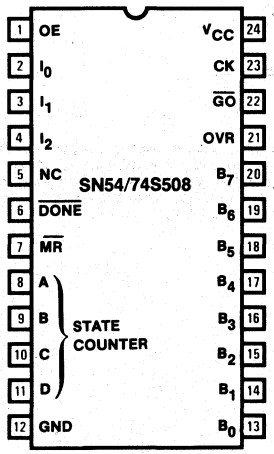
Ordering Information

PART NUMBER	PACKAGE	TEMPERATURE
SN54S508	D24	Military
SN74S508	D24	Commercial

Logic Symbol



Pin Configuration



2175 Mission College Blvd. Santa Clara, CA 95054-1592 Tel: (408) 970-9700 TWX: 910-338-2376 TWX: 910-338-2374
Monolithic Memories MMI

INSTRUCTION SEQUENCE				OPERATION	CLOCK CYCLES
ARITHMETIC OPERATIONS					
			0	X1 · Y	5
			1	-X1 · Y	5
			2	X1 · Y + K_Z, K_W	5
			3	-X1 · Y + K_Z, K_W	5
			4	K_Z, K_W/X1	13
		5/6	0	X · Y	6
		5/6	1	-X · Y	6
		5/6	2	X · Y + K_Z, K_W	6
		5/6	3	-X · Y + K_Z, K_W	6
		5/6	4	K_W/X	14
		5/6	5	K_Z/X	14
	5/6	6	0	X · Y + Z	7
	5/6	6	1	-X · Y + Z	7
	5/6	6	2	X · Y + $K_Z \cdot 2^{-8}$	7
	5/6	6	3	-X · Y + $K_Z \cdot 2^{-8}$	7
	5/6	6	4	Z, W/X	15
	5/6	6	5	Z/X	15
5/6	6	6	0	X · Y + Z, W	8
5/6	6	6	1	-X · Y + Z, W	8
5/6	6	6	2	X · Y + W_{sign}	8
5/6	6	6	3	-X · Y + W_{sign}	8
5/6	6	6	4	W/X	16
5/6	6	6	5	W_{sign}/X	16
5/6	6	6	6	(See Note 9 below)	—
5/6	5/6	6	7	Load X, Load Z, Load W, Clear Z	3
READING OPERATIONS					
			7	Read Z	1
		7	7	Read Z, W	2
	7	7	7	Read Z, W, Z	3
7	7	7	7	Read Z, W, Z, W	4
		5	7	Round, then Read Z	2
	5	7	7	Round, then Read Z, W	3

NOTES:
1. X,Y are input multiplier and multiplicand.
2. X1 is the previous contents of the first rank of the X register, (either the old X or a new X).
3. Fractional or integer arithmetic is specified by having the next-to-the-last operand loaded using a 5 or 6 instruction respectively. All rows beginning with "5/6" in effect represent *two* instructions. 5 does fractional arithmetic and 6 does integer arithmetic.
4. Z, W is a double-precision number. Z is the most significant half. Z, W represents addend upon input, and product (or accumulated sum) after multiplication.
5. K_Z, K_W represents previous accumulator contents. K_Z is the most-significant half.
6. W_{sign} is a single-length signed number, with sign extension.
7. Maximum clock cycle = 167 ns for a 6-MHz clock.
8. If n instruction codes are shown at the left under "instruction sequences," the number of clock cycles at the right is n+4 for multiplication and n+12 for division.
9. The code "5/6 6 6 6" represents an incomplete operation since it leaves the 'S508 in state 1 rather than in state 0, 8, or 10

Figure 1. 'S508 Instruction Set (Partial List)

SUMMARY OF SIGNALS/PINS	
B_7-B_0	Bidirectional data bus inputs/outputs
I_2-I_0	Instruction (sequential control) input
A, B, C, D	Internal-state-counter outputs
CK	Clock pulse input
$\overline{GO}$	Chip activation input
OE	Output enable input
$\overline{MR}$	Master reset input
OVR	Arithmetic overflow output
$\overline{DONE}$	Arithmetic-operation completion output

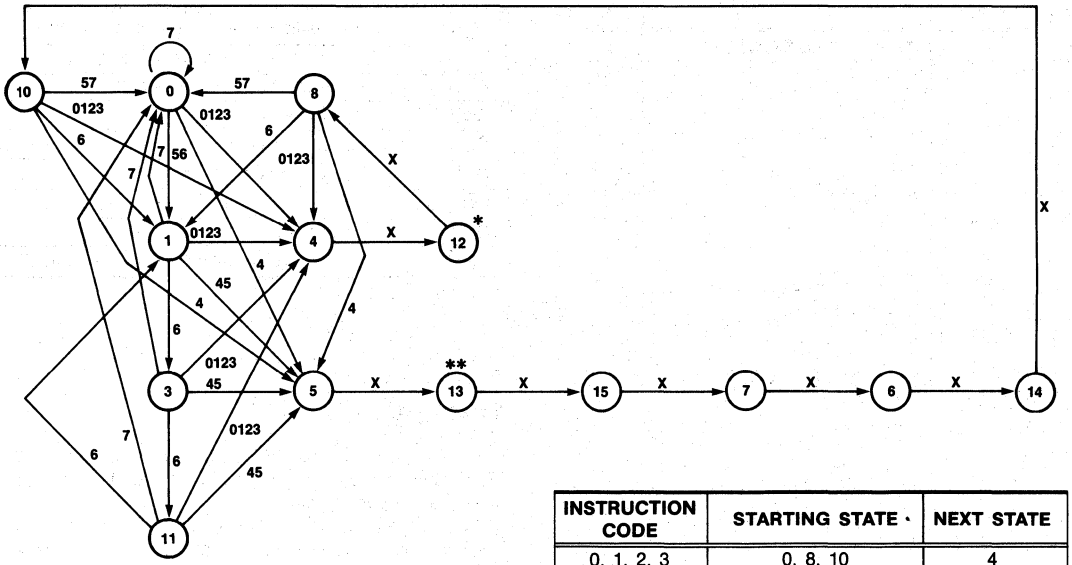
Description (continued)

The 'S508 device uses standard low-power Schottky technology, requires a single +5V power supply, and is fully TTL compatible. Bus inputs require at most 250 μA input current, and control and clock inputs require at most 1 mA input current. Bus outputs are three-state, and are capable of sinking 8 mA at the low logic level. The 'S508 is available in both commercial-temperature and military-temperature ranges, in a 600-mil 24-pin dual-in-line ceramic package.

Device Operation

The 'S508 contains four 8-bit working registers. Y is the multiplier register; X is the multiplicand and divisor register; W is the least-significant half of a double-length accumulator, and holds the least-significant half of the product after a multiplication operation, or the remainder after a division operation; and Z is the most-significant half of this same accumulator. In addition to these registers, there is a high-speed arithmetic unit which performs addition, subtraction, and shifting steps in order to accomplish the various arithmetic operations; a loading sequencer; and a PLA control network.

Operands are loaded into the working registers in time sequence at each clock period, under the control of this sequencer. The chip-activation signal $\overline{GO}$ must be LOW in order to begin the loading process and continue to the next step in the loading operation. If $\overline{GO}$ is continually held HIGH, the 'S508 remains in a wait state with its outputs held in their high-impedance states, so that the other devices attached to the bus may drive it. In this condition, the 'S508 does not respond to any codes on its instruction inputs; in effect, it does not "wake up" until $\overline{GO}$ goes LOW. Also, $\overline{GO}$ may change only when the clock input CK is HIGH. After all of the operands are loaded, the 'S508 jumps to the multiply routine, or to the divide routine, and performs the required operations as indicated in Figure 1. After 5 clock periods for a simple multiply or 13 clock periods for a simple divide, for example, the device is ready to place the result on the bus in time sequence.



INSTRUCTION CODE	STARTING STATE	NEXT STATE
0, 1, 2, 3	0, 8, 10	4
4	0, 8, 10	5
5	0	1
5, 7	8, 10	0
6	0, 8, 10	1
7	0, 8, 10	0

*Loop 3 times for multiplication.
**Loop 6 times for fractional division, or 7 times for integer division.

KEY:

The numbers inside the circles indicate the *state* of the 'S508 multiplier/divider. These states are represented by a four-bit state counter, where A is the least-significant bit of this state counter and D is the most-significant bit. These four bits are available externally on the 'S508.

The next state of the 'S508 is a function of the present state and the instruction lines. For example if the 'S508 is at state 0 and the instruction is 0, 1, 2, or 3, then the next state is state 4 (multiply instruction); if the instruction is 4, the next state is state 5 (divide instruction); and so forth. The instructions which take the 'S508 from one state to another are indicated by the numbers written next to the state-transition path lines. "0123," for instance, implies that *any* of instructions 0, 1, 2, or 3 will take the 'S508 along the path marked "0123."

"X" next to a path implies that the path will be followed regardless of the value of the instruction inputs at that time. In other words, for the purpose of state transitions, X means "don't care." There are cases, however, where the particular instruction used may affect when the contents of the registers are available on the bus — see Figures 9 and 10 for contrasting examples of how this effect operates.

Figure 2. Transition Diagram for the 'S508 Multiplier/Divider

Three instruction inputs I_2, I_1, I_0, which may change only when the clock input CK is HIGH, select the required function and drive the sequencer from state to state. Thus, the action of the multiplier/divider at any clock period is a function of the machine state and the state of the control inputs. Figure 2 shows the multiply/divide state table, and all possible operations. After a Read or Round operation, the machine is driven back to state 0, and a new sequence of arithmetic operations is assumed. If a chain operation is being performed, such as accumulation of products, state 0 is bypassed, and loading of an operand or jumping to the next arithmetic operation occurs at the end of the previous arithmetic operation — at state 8 for a multiplication instruction, or at state 10 for a division instruction.

Register X is a dual-rank register, which allows the loading of an operand X during the multiplication or division process. If the machine enters the loading sequence and a new X operand has not been loaded, then the machine proceeds with the previously-loaded X, denoted in this text as "X1." This loading-while-processing capability allows a cycle to be saved during "chained" calculations, and also allows multiplication and division by a constant. (See Figure 13). (continued next page)

Figures 3 and 4 show the codes and durations for the 41 different possible arithmetic operations. These operations can be concatenated in strings to perform complicated 2s-complement arithmetic operations at high-speed. Rounding and reading of results can be performed after any operation. Figure 5 is a block diagram of the 'S508 8x8 Multiplier/Divider.

(continued page after next)

OPERATION		TIME-SLOT 1	2	3	4	5	6	7	8
X1 · Y	INS CODE BUS	0 Y	MULTIPLY						
-X1 · Y	INS CODE BUS	1 Y	MULTIPLY						
X1 · Y + K_Z, K_W	INS CODE BUS	2 Y	MULTIPLY						
-X1 · Y + K_Z, K_W	INS CODE BUS	3 Y	MULTIPLY						
X · Y	INS CODE BUS	5/6 0 X Y	MULTIPLY						
-X · Y	INS CODE BUS	5/6 1 X Y	MULTIPLY						
X · Y + K_Z, K_W	INS CODE BUS	5/6 2 X Y	MULTIPLY						
-X · Y + K_Z, K_W	INS CODE BUS	5/6 3 X Y	MULTIPLY						
X · Y + Z	INS CODE BUS	5/6 6 0 X Z Y	MULTIPLY						
-X · Y + Z	INS CODE BUS	5/6 6 1 X Z Y	MULTIPLY						
X · Y + $K_Z \cdot 2^{-8}$	INS CODE BUS	5/6 6 2 X — Y	MULTIPLY						
-X · Y + $K_Z \cdot 2^{-8}$	INS CODE BUS	5/6 6 3 X — Y	MULTIPLY						
X · Y + Z, W	INS CODE BUS	5/6 6 6 0 X Z W Y	MULTIPLY						
-X · Y + Z, W	INS CODE BUS	5/6 6 6 1 X Z W Y	MULTIPLY						
X · Y + W_{sign}	INS CODE BUS	5/6 6 6 2 X — W Y	MULTIPLY						
-X · Y + W_{sign}	INS CODE BUS	5/6 6 6 3 X — W Y	MULTIPLY						

NOTES: 1) X1 is the previous contents of the first rank of the X register (either old X or a new X).
2) $K_Z \cdot 2^{-8}$ is a single-length signed number comprising the most-significant half of the previous double-length product and here gets added in at the least-significant end of the new result.
3) W_{sign} is a single-length signed number, with sign-extension as needed.
4) Fractional or integer arithmetic is specified by having the next-to-last operand loaded using a 5 or 6 instruction respectively. All rows beginning with "5/6" in effect represent *two* instructions. 5 does fractional arithmetic and 6 does integer arithmetic.

Figure 3. Multiplication Codes and Times for 8x8 Multiplication in the 'S508

OPERATION		TIME-SLOT 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
K_Z, K_W/X_1	INS CODE	4	DIVIDE											1			
	BUS	—															
K_W/X	INS CODE	5/6	4	DIVIDE											1		
	BUS	X	—														
K_Z/X	INS CODE	5/6	5	DIVIDE											1		
	BUS	X	—														
Z, W/X	INS CODE	5/6	6	4	DIVIDE											1	
	BUS	X	Z	W													
Z/X	INS CODE	5/6	6	5	DIVIDE											1	
	BUS	X	Z	—													
W/X	INS CODE	5/6	6	6	4	DIVIDE											1
	BUS	X	—	W	—												
W_{sign}/X	INS CODE	5/6	6	6	5	DIVIDE											1
	BUS	X	0	W	—												

NOTES: 1) X1 is the previous contents of the first rank of the X register (either old X or a new X).
2) Fractional division divides a 16-bit 2s-complement number in 1 clock period less than integer division.
3) W_{sign} is a single-length signed number, with sign-extension as needed.
4) Division operation W_{sign}/X requires that the Z register be initialized with all-zero contents at the time Z is loaded.
5) Fractional or integer arithmetic is specified by having the operand loaded using a 5 or 6 instruction respectively. All rows beginning with "5/6" in effect represent *two* instructions, one of which does fractional arithmetic and one of which does integer arithmetic.

Figure 4. Division Codes and Time for 16/8 Division in 'S508

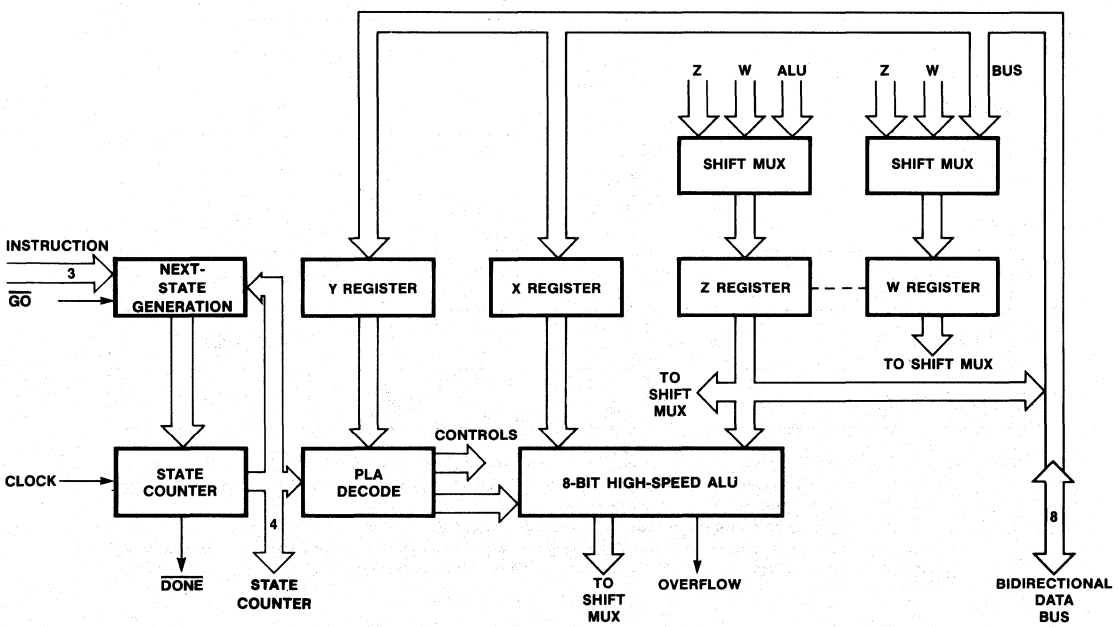


Figure 5. Internal Architecture of the 'S508

Multiplication

The 'S508 provides 2s-complement 8-bit multiplication, and can also accumulate previously-generated double-length products. No time penalty is incurred for accumulation, since the machine accumulates while the multiplication operation is proceeding. In addition to accumulation, the device can add into a product either a single-length or a double-length number. It can also use a previously-loaded operand as a constant, so that constant multiplication and accumulation is possible.

One key feature is the ability to perform both positive multiplications and negative multiplications, again without any speed penalty. This feature allows complex-arithmetic multiplications to be programmed with very little overhead. Another important feature is the ability to work with either fractions or integers.

Division

The 'S508 also provides a range of division operations. A double-length number in Z,W is divided by X; the result Q is stored in Z, and the remainder R in W. Again all numbers are in the 2s-complement number representation, with the most significant bit of an operand (whether single-length or double-length) having a negative weight. In order to facilitate repeated division, with the multiple-length quotient always keeping the same sign, the remainder is always the same sign as the dividend. Fractional or integer operation is possible, and division and multiplication operations can be concatenated. For example, the operations (AxB)/C,(A+B)/C can easily be performed. The dividend can be any previously-generated result — product, quotient, or remainder; or it may be a double-length or single-length signed operand.

Reading Results

The result of an arithmetic operation, or of a string of operations, can be read onto the 8-bit bus if the machine is at the end of an operation or at the start of a new sequence. The read operation requires that the $\overline{GO}$ signal be held LOW so that the information is read out onto the bidirectional bus, when code 7 is specified. (See Figure 6.) Since there is a double-length accumulator Z,W, reading can take two cycles. First, register Z is read. After another clock has been received, if code 7 is still present, the least-significant half of the product from the W register is placed on the bus, or likewise the remainder if a division operation had been performed.

If the 'S508 is instructed to perform a read operation during the loading sequence, then the sequence is broken and the machine is forced back to state 0 ready to start the sequence again. Continual read operations at state 0 just swap the contents of register Z and W.

The 'S508 has a direct master reset input $\overline{MR}$. Alternatively, initialization of the 'S508 can also easily be performed by continually presenting instruction code 7, which after a maximum of 13 clock periods forces the machine back to state 0.

Integer and Fractional Arithmetic

The 'S508 can work with either fractional or integer number representations. When working with integers, all numbers are scaled from the least-significant end and the least-significant bit is assumed to have a weight of 2^0. For integer multiplication, accumulation, and division, all numbers are scaled from this least-significant weight, and results are correct if interpreted in this manner. The double-length register Z,W can therefore hold numbers in the range -2^{15} to $+2^{15}-1$; the operands X and Y, and single-length results, are in the range -2^7 to $+2^7-1$.

When working with fractions, the machine automatically performs scaling so that input operands and results have a consistent format. All numbers in the fractional representation are scaled from the most significant end, which has a weight of -2^0 *(negative)*. The binary point is one place to the right of this most-significant bit, so that the next bit has a weight of 2^{-1}. The double-length register Z,W therefore holds numbers in the range -1 to $+1-2^{-15}$ and the operands X and Y and single-length results are in the range -1 to $+1-2^7$. Since automatic scaling occurs, the product of two numbers always has the least-significant bit as a 0, unless an accumulation is performed with the least-significant bit being a 1.

During a chain operation with the partial results not being read onto the bus, the 'S508 will stay in either the fractional or integer mode. At the start of a sequence of operations, fractional or integer operation is designated by loading operands using instruction code 5 or instruction code 6 respectively.

Mixed fractional and integer arithmetic is also possible, by redefining the weight of the least-significant or most-significant bits. However, care must be exercised, due to the automatic scaling feature, when fractional arithmetic is programmed.

Rounding

Rounding can be performed on the result of a multiplication or division. Generally rounding would only be called out during fractional operation, but nothing in the 'S508 precludes forming a rounded result during integer arithmetic.

Rounding for multiplication provides the best single-length most-significant half of the product. Rounding occurs at the end of a multiplication, and is performed instead of a Load or Read operation when a code 5 is specified, instead of a code 7, to get from state 8 or state 10 back to state 0. (See Figure 2; also, note that this mode of operation precludes "stealing" a cycle according to the method illustrated in Figure 9.) The 'S508 looks at the most-significant bit of the least-significant half of the product W_7, and adds 1 to the most-significant half of the product at the least-significant end if W_7 is a 1. After the operation, the 'S508 is in state 0, so that the rounded product can be read, and the W register is clear.

Rounding for division is performed by forcing the least-significant bit of the quotient in Z to a 1 unless the division is exact (remainder is zero). This method of rounding causes a slightly higher variance in the result than having an additional iterative division operation, but is considerably easier to perform. Again, after rounding the 'S508 goes to state 0, so that a read operation can be performed, and the W register is clear.

10

Overflow

The 'S508 has an overflow output OVR which is cleared prior to each operation, and is set during an operation if the product or quotient goes outside the normally-accepted range.

For multiplication, overflow can only occur if the most negative number in the operand range is used: $(-1)x(-1)=+1$, which cannot be held in the 'S508's internal registers. Overflow can more easily occur during either positive or negative accumulation of products. For fractional arithmetic, if the product or accumulation goes outside the range of -1 to $+1-2^{-15}$, then the overflow flipflop will be set.

Overflow may also occur during division if the quotient goes outside the generally-accepted number range of -1 to $+1-2^{-7}$ during fractional operation. This would occur if the divisor is less than the dividend, or equal to the dividend if a positive quotient is being generated. For integer arithmetic the numbers must be scaled by 2^7.

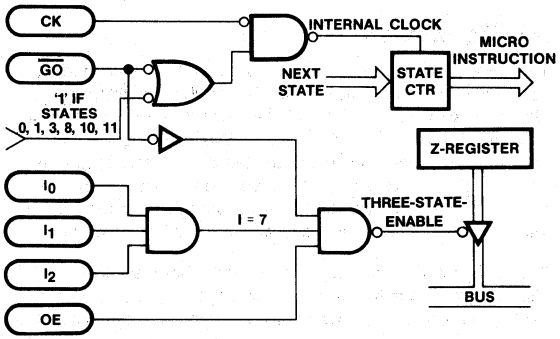


Figure 6. 'S508 Internal Circuitry of "$\overline{GO}$" Line and Three-State-Enable

During the states 0, 1, 2, 3, 8, 10 and 11 if the "GO" line ($\overline{GO}$) is held at logic HIGH then the machine will be in a wait state until $\overline{GO}$ goes to logic LOW.

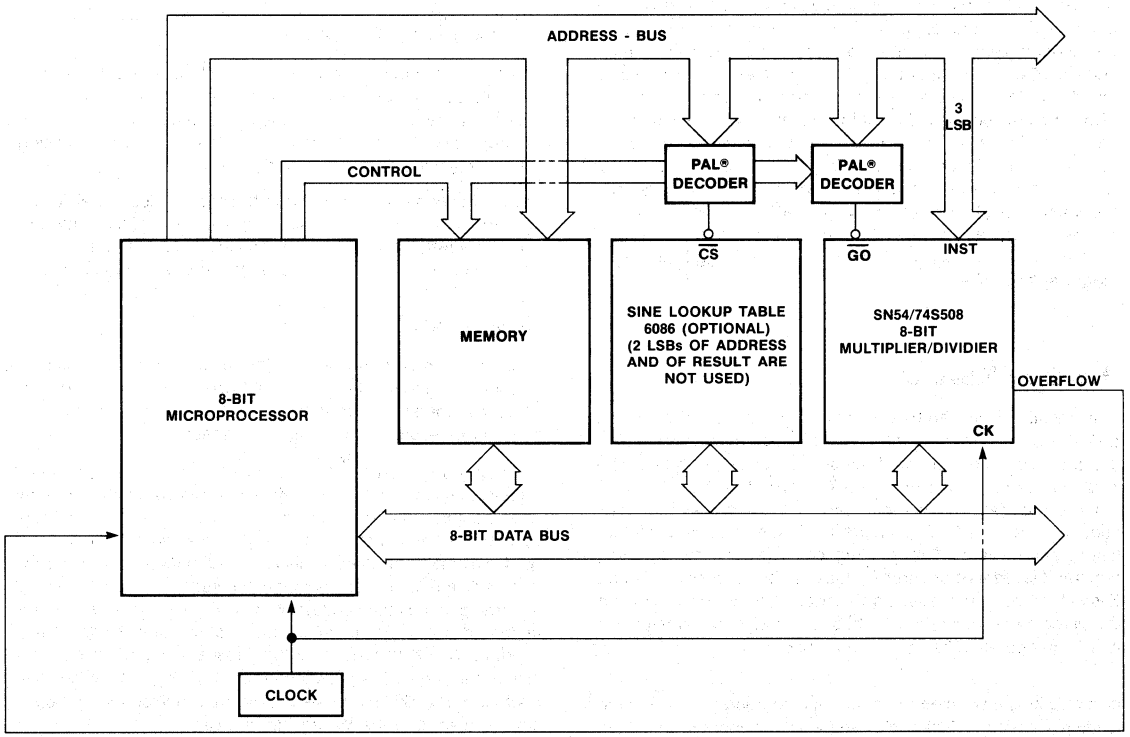


Figure 7. Interfacing the 'S508 to an 8-bit Microprocessor

Figure 7 shows the block diagram of a minimum 8-bit microprocessor system with its arithmetic capabilities enhanced by the use of a 'S508 8x8 multiplier/divider. The relatively small number of instruction lines (only 3) of the 'S508 provides a unique way to control the multiplier/divider. As may be seen from Figure 7, these three instruction lines are assigned to the three least-significant bits (LSBs) of the address bus, while the remaining address bits are decoded by a Programmable Array Logic (PAL®) circuit to determine when the multiplier/divider is selected. For example, suppose the 'S508 is assigned address 100; then any address in the range of 100-107 will enable the 'S508 (i.e., the $\overline{GO}$ line is LOW). Thus, if the address is 100 the 'S508 instruction is 0; if the address is 106 the 'S508 instruction is 6; and so forth.

Absolute Maximum Ratings

Supply voltage V_{CC} ..... 7.0 V
Input voltage ..... 7.0 V
Off-state output voltage ..... 5.5 V
Storage temperature ..... −65° to +150°C

Operating Conditions

SYMBOL	PARAMETER	FIGURE	MILITARY			COMMERCIAL			UNIT
			MIN	TYP	MAX	MIN	TYP	MAX	
V_{CC}	Supply voltage		4.5	5	5.5	4.75	5	5.25	V
T_A	Operating free-air temperature		−55		125†	0		75	°C
f_{MAX}	Clock frequency	8	5			6			MHz
t_{CWP}	Positive clock pulse width	8	90			70			ns
t_{CWN}	Negative clock pulse width	8	60			50			ns
t_{BS}	Bus setup time for inputting data *	8	60			50			ns
t_{BH}	Bus hold time for inputting data *	8	45			35			ns
t_{INSS}	Instruction, $\overline{GO}$ setup time	8	10			10			ns
t_{INSH}	Instruction, $\overline{GO}$ hold time	8	20			20			ns

* During operations when the bus is being used to input data.
† Case temperature.

Electrical Characteristics Over Operating Conditions

SYMBOL	PARAMETER	TEST CONDITIONS		MIN	TYP	MAX	UNIT
V_{IL}	Low-level input voltage					0.8	V
V_{IH}	High-level input voltage			2			V
V_{IC}	Input clamp voltage	V_{CC} = MIN I_I = −18mA				−1.5	V
I_{IL}	Low-level input current	V_{CC} = MAX V_I = 0.5V	B_7–B_0			−250	μA
			All other inputs			−1	mA
I_{IH}	High-level input current	V_{CC} = MAX V_I = 2.4V				250	μA
I_I	Maximum input current	V_{CC} = MAX V_I = 5.5V				1	mA
V_{OL}	Low-level output voltage	V_{CC} = MIN I_{OL} = 8mA			0.3	0.5	V
V_{OH}	High-level output voltage	V_{CC} = MIN I_{OH} = −2mA		2.4			V
I_{OS}	Output short-circuit current*	V_{CC} = MAX V_O = 0V		−10		−90	mA
I_{CC}	Supply current	V_{CC} = MAX	SN54S508		300	400	mA
			SN74S508		300	380	

* Not more than one output should be shorted at a time, and the duration of the short-circuit should not exceed one second.

Switching Characteristics

Over Operating Conditions

SYMBOL	PARAMETER		FIGURE	MILITARY			COMMERCIAL			UNIT
				MIN	TYP	MAX	MIN	TYP	MAX	
t_{BO}	Bus output delay for outputting data*		8		70	120		70	95	ns
t_{PXZ}	Output disable delay	From I_2–I_0 to bus			40	70		40	65	ns
		From OE, $\overline{GO}$ to bus			20	50		20	40	
t_{PZX}	Output enable delay	From I_2–I_0 to bus			45	90		45	80	ns
		From OE, $\overline{GO}$ to bus			25	55		25	45	
t_{OVR}	Overflow output delay from CK		8		70	120		70	95	ns
t_{DN}	$\overline{DONE}$ output delay		8		30	90		30	70	ns

* During operations when the bus is being used to output data.

10

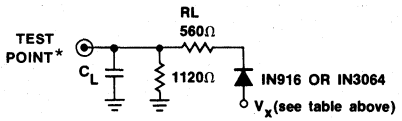
AC Test Conditions

Inputs 0 V_{LOW}, 3 V_{HIGH}. Rise and fall time 1-3 ns from 1 V to 2 V. Measurements are made from 1.5 V_{IN} to 1.5 V_{OUT}, except that t_{PXZ} is measured by a delta in the outputs of 0.5 V from V_{OL} or V_{OH} respectively.

Timing

Timing waveforms are shown in Figure 8. Specific instruction timing examples are shown in Figures 9 through 13.

Test Load

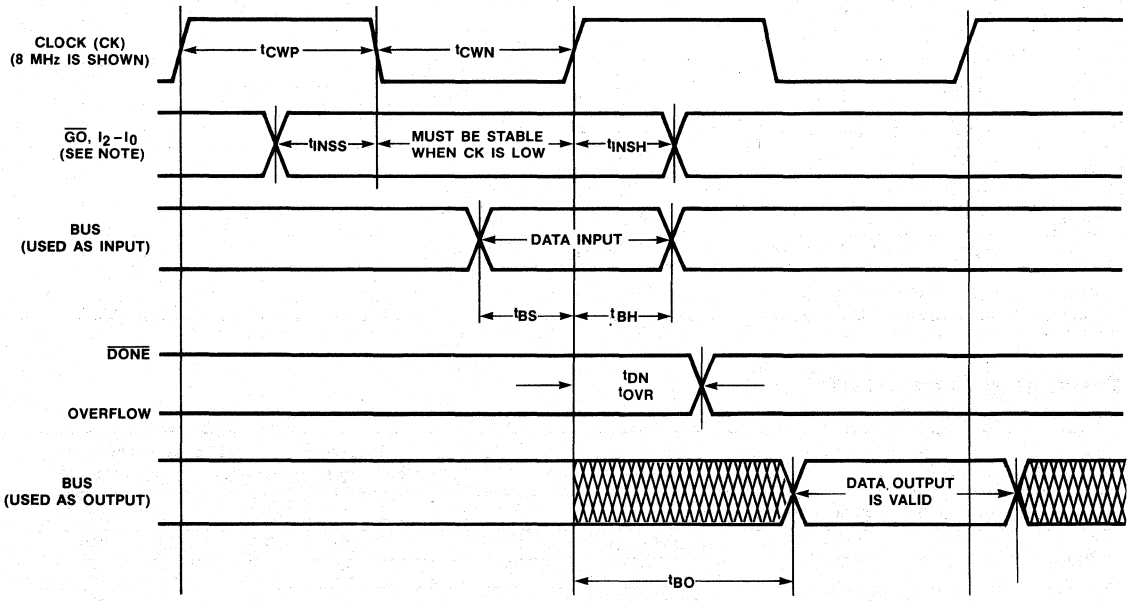


* The "TEST POINT" is driven by the output under test, and observed by instrumentation.

Test Waveforms

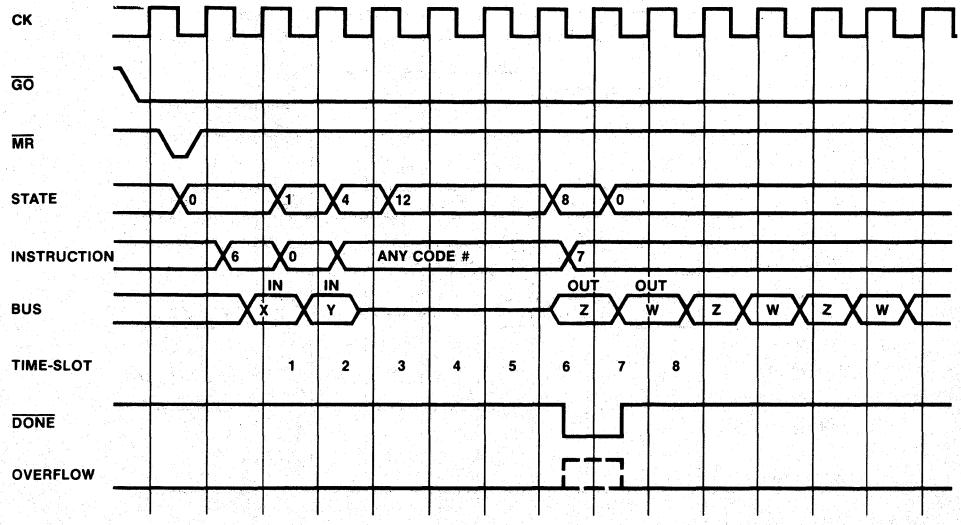
TEST	V_X*		OUTPUT WAVEFORM — MEAS. LEVEL
All t_{PD}	5.0V		V_{OH}, V_{OL}, 1.5V
t_{PXZ}	t_{PHZ}	t_{PLZ}	V_{OH}, 0.5V, 0.5V, V_{OL}, 2.8V, 0.0V
	0.0V	5.0V	
t_{PZX}	t_{PZH}	t_{PZL}	2.8V, 0.0V, 1.5V, V_{OH}, V_{OL}
	0.0V	5.0V	

* At diode: see "Test Load" figure at left.



NOTE: $\overline{GO}$ and I_2-I_0 can change only when CK is high.

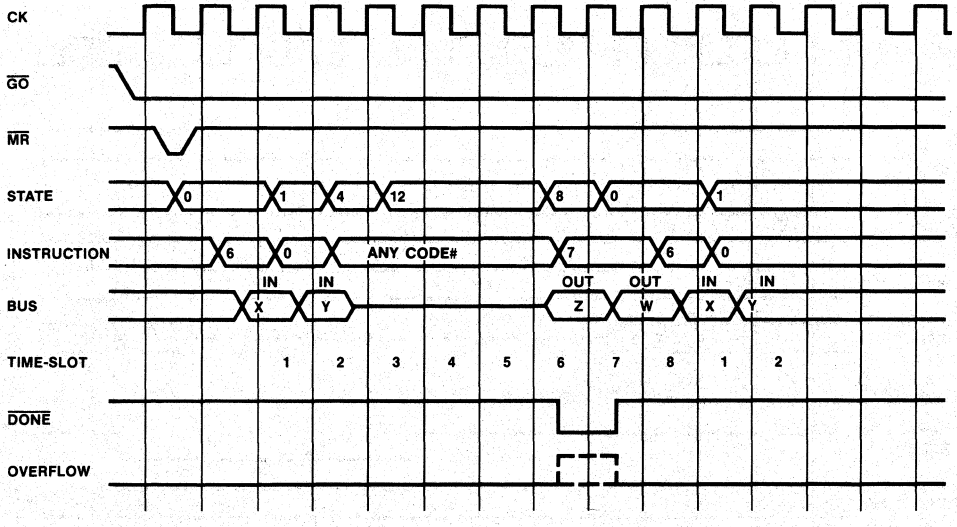
Figure 8. Timing Diagram of the 'S508



NOTES: Register Z is read at the same time that the "$\overline{\text{DONE}}$" signal is set. If the instruction remains at code 7 after time-slot 7, the contents of registers Z and W are swapped each cycle.

"Any code" means code 0 through 7. However code 6 will load a new value of X, and code 7 will cause the 'S508 to attempt to drive the data bus.

Figure 9. Instruction Timing Example No. 1: Load X, Load Y, Multiply, Read W. By Presenting Code 7 on the Instruction Lines During the Last Multiply Cycle (State 8), the Results May Be Read During Time Slots 6 and 7

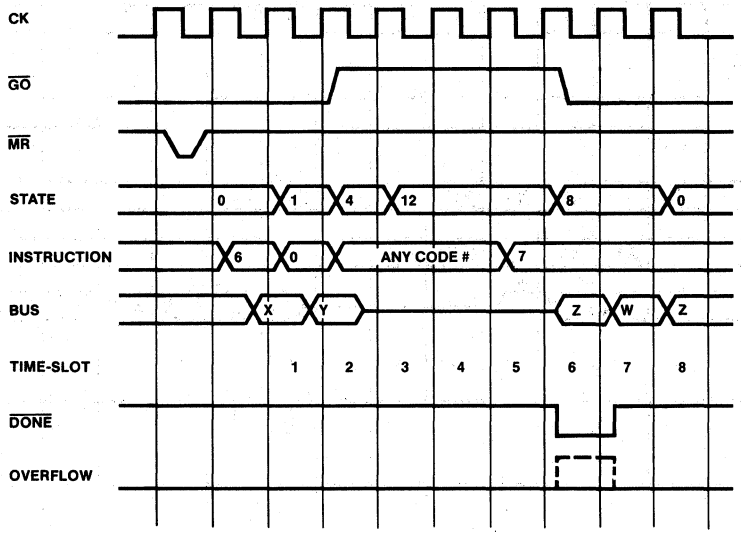


NOTES: The instruction lines may be changed only when CK is high.

#"Any code" means code 0 through code 7.

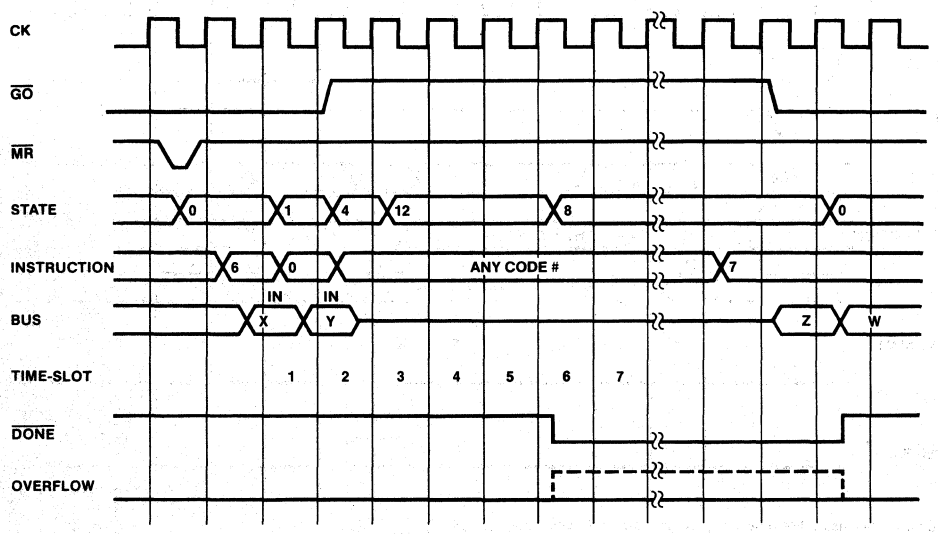
Code 6 may be used here since a new X explicitly gets loaded for the next multiply operation. However, code 7 will cause the 'S508 to attempt to drive the data bus.

Figure 10. Instruction Timing Example No. 2: Repeat: "Load X, Load Y, Multiply, Read Z, Read W"



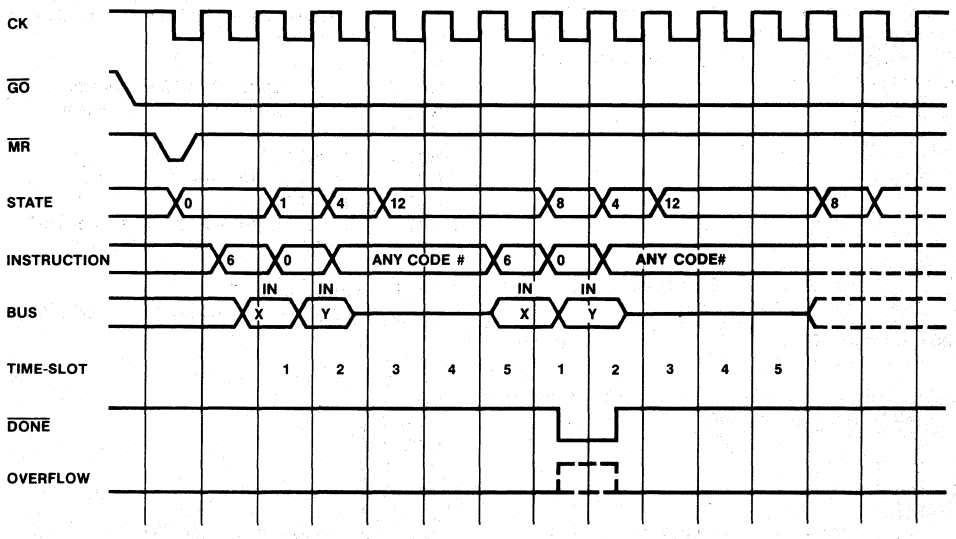
NOTE: If code 7 is given (instead of code 0 through 6), the first data that is read from the bus after the $\overline{DONE}$ signal is set (time-slot 7) is W and not Z. However, Z is read at time-slot 8.
#"Any code" means code 0 through code 7.

Figure 11. Instruction Timing Example No. 3: Load X, Load Y, Multiply, Read Z, Read W. This timing diagram corresponds to Table 1. Only after the $\overline{DONE}$ signal is set (after four clock pulses of the operation cycles), the result is read — Z during time-slot 7, and W during time-slot 8



NOTE:
"Any code" means code 0 through code 7. Code 6 or code 7 may be used here. Since $\overline{GO}$ is HIGH, no new X can be loaded and the 'S508 can not attempt to drive the bus.

Figure 12. Instruction Timing Example No. 4: Load X, Load Y, Multiply, Wait, Read Z, Read W



NOTES: This sequence of operations is suitable for use when reading is to be done only at the very end of the operation sequence. The new X value is loaded during the time that the previous multiplication is being performed. See Programming Example #3 for $\sum_{i=1}^{N} X_i \cdot Y_i$

#"Any code" means code 0 through code 7.
†Code 6 allows loading of a new X value in state 12 and it takes the 'S508 to state 8. In state 8, Y is loaded via instruction 2 and the multiply-accumulate operation is initiated.

Figure 13. Instruction Timing Example No. 5: Sum of Products

10

Programming Examples

In the following examples assume that each line with a separate instruction corresponds to one clock pulse. Instruction codes are 0, 1, 2, 3, 4, 5, 6, 7 and x according to the usage explained in the key to Figure 2.

Programming Example 1

Calculating X·Y (A·B)

INST 6	X ← A
INST 0	Y ← B
INST X	MULT
INST X	MULT
INST X	MULT
INST 7	MULT and READ Z = 8 MSB OF (A·B)
INST 7	READ W = 8 LSB OF (A·B)

Programming Example 2

Calculating X1 · Y (A · C)

X1 is a previous multiplier value. It was previously loaded (in example 1) with A.

INST 0	Y ← C
INST X	MULT
INST X	MULT
INST X	MULT
INST 7	MULT and READ Z = 8 MSB OF (A·C)
INST 7	READ W = 8 LSB OF (A·C)

Programming Example 3

Calculating $\sum_{i=1}^{N} X_i \cdot Y_i$ (A·B + C·D + E·F + ...)

In this case we read only after N multiplications. A new X_{i+1} is loaded during the multiplication process for $X_i Y_i$.

Assume N = 3.

The sequence of instructions and operations for calculating $\sum_{i=1}^{3} X_i \cdot Y_i$ is: (A·B + C·D + E·F)

N = 1	INST 6	X ← A
	INST 0	Y ← B
	INST X	MULT } Perform A·B
	INST X	MULT
	INST X	MULT
N = 2	INST 6	MULT and LOAD X ← C
		Z ← 8 MSB of (A·B)
		W ← 8 LSB of (A·B)
	INST 2	Y ← D
	INST X	MULT } Perform C·D + (K_Z, K_W)
	INST X	MULT
	INST X	MULT
N = 3	INST 6	MULT and LOAD X ← E
		Z ← 8 MSB of (C·D + A·B)
		W ← 8 LSB of (C·D + A·B)
	INST 2	Y ← F
	INST X	MULT } Perform E·F + (K_Z, K_W)
	INST X	MULT
	INST X	MULT
READ Z	INST 7	MULT and
		READ Z = 8 MSB of (E·F + C·D + A·B)
READ W	INST 7	READ W = 8 LSB of (E·F + C·D + A·B)

Programming Example 4

Multiplication plus a constant (A·B + Constant (16 bits))

Assume that the constant is a 16-bit 2s-complement number.

INST 6	X ← A	
INST 6	Z ← C	LOAD 8 MSB of constant
INST 6	W ← D	LOAD 8 LSB of constant
INST 0	Y ← B	
INST X	MULT	} Perform A·B + (Z, W)
INST X	MULT	
INST X	MULT	
INST 7	MULT and READ	Z = 8 MSB of (A·B + (C, D))
INST 7	READ	W = 8 LSB of (A·B + C, D)

Programming Example 5

Dividing a 16-bit number by an 8-bit number ((B, C)/A)

INST 6	X ← A	
INST 6	Z ← B	
INST 4	W ← C	
INST X		} Perform Division $\frac{(Z, W)}{X}$
INST X		
INST X		
INST X		
INST X		
INST X		
INST X		
INST X		
INST X		
INST X		
INST X		
INST 7	DIVIDE and READ the quotient Z =	$\frac{(B, C)}{A}$
INST 7	READ the remainder W of	$\frac{(B, C)}{A}$

16x16 Multiplier/Divider
SN74S516

Features/Benefits

- **Co-processor for enhancing the arithmetic speed of all present 16-bit and 8-bit microprocessors**
- **Bus-oriented organization**
- **24-pin package**
- **16/16 or 32/16 division in less than 3.5 μsec**
- **16x16 multiplication in less than 1.5 μsec**
- **28 different multiplication instructions such as "fractional multiply and accumulate"**
- **13 different divide instructions**
- **Self-contained and microprogrammable**

Description

The SN74S516 ('S516) is a bus-organized 16x16 Multiplier/Divider. The device provides both multiplication and division of 2s-complement 16-bit numbers at high speed. There are 28 different multiply options, including: positive and negative multiply, positive and negative accumulation, multiplication by a constant, and both single-length and double-length addition in conjunction with multiplication. 13 different divide options allow single-length or double-length division, division of a previously-generated result, division by a constant, and continued division of a remainder or quotient.

The 'S516 is a time-sequenced device requiring a single clock. It loads operands from, and presents results to, a bidirectional 16-bit bus. Loading of the operands, reading of the results, and sequential control of the device is performed by a 3-bit instruction field.

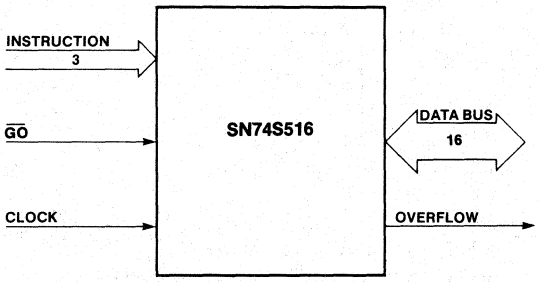
The 'S516 has the additional feature that operands and results can be either integers or fractions; when it deals with fractions, automatic scaling occurs. Results can be rounded if required, and an Overflow output indicates whenever a result is outside the normally-accepted number range.

For a simple multiplication of two operands the device takes nine clock periods — one for initialization, and eight for the actual multiplication. A realistic clock period is 167 ns, which gives a multiplication time of 1333 ns typical for 16x16 multiplication, plus 167 ns additionally for initialization, or 1500 ns in all. More complex multiplications will take additional clock periods for loading the additional operands. A simple division operation requires 16 + 4 = 20 clock periods for a typical time of 3.333 ns (32 bits/16 bits), also plus 167 ns for initialization, or 3500 ns in all.

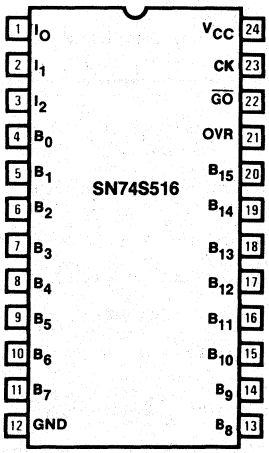
Ordering Information

PART NUMBER	PACKAGE	TEMPERATURE
SN74S516	24T	Commercial

Logic Symbol



Pin Configuration



10

2175 Mission College Blvd. Santa Clara, CA 95054-1592 Tel: (408) 970-9700 TWX: 910-338-2374
TWX: 910-338-2376
Monolithic Memories MMI

INSTRUCTION SEQUENCE				OPERATION	CLOCK CYCLES
ARITHMETIC OPERATIONS					
			0	X1 · Y	9
			1	-X1 · Y	9
			2	X1 · Y + K_Z, K_W	9
			3	-X1 · Y + K_Z, K_W	9
			4	K_Z, K_W/X1	21
		5/6	0	X · Y	10
		5/6	1	-X · Y	10
		5/6	2	X · Y + K_Z, K_W	10
		5/6	3	-X · Y + K_Z, K_W	10
		5/6	4	K_W/X	22
		5/6	5	K_Z/X	22
	5/6	6	0	X · Y + Z	11
	5/6	6	1	-X · Y + Z	11
	5/6	6	2	X · Y + $K_Z \cdot 2^{-16}$	11
	5/6	6	3	-X · Y + $K_Z \cdot 2^{-16}$	11
	5/6	6	4	Z, W/X	23
	5/6	6	5	Z/X	23
5/6	6	6	0	X · Y + Z, W	12
5/6	6	6	1	-X · Y + Z, W	12
5/6	6	6	2	X · Y + W_{sign}	12
5/6	6	6	3	-X · Y + W_{sign}	12
5/6	6	6	4	W/X	24
5/6	6	6	5	W_{sign}/X	24
5/6	6	6	6	(See Note 9 below.)	—
5/6	6	6	7	Load X, Load Z, Load W, Clear Z	4
	5/6	6	7	Load X, Load Z, Read Z	3
READING OPERATIONS					
			7	Read Z	1
		7	7	Read Z, W	2
	7	7	7	Read Z, W, Z	3
7	7	7	7	Read Z, W, Z, W	4
		5	7	Round, then Read Z	2
	5	7	7	Round, then Read Z, W	3

NOTES:
1. X,Y are input multiplier and multiplicand.
2. X1 is the previous contents of the first rank of the X register (either the old X or a new X).
3. Fractional or integer arithmetic is specified by having the next-to-the-last operand loaded using a 5 or 6 instruction respectively. All rows beginning with "5/6" in effect represent *two* instructions. 5 does fractional arithmetic and 6 does integer arithmetic.
4. Z, W is a double-precision number. Z is the most significant half, Z, W represents addend upon input, and product (or accumulated sum) after multiplication.
5. K_Z, K_W represents previous accumulator contents. K_Z is the most-significant half.
6. W_{sign} is a single-length signed number, with sign extension.
7. Maximum clock cycle = 167 ns for an 6-MHz clock.
8. If n instruction codes are shown at the left under "instruction sequences," the number of clock cycles at the right is n+8 for multiplication and n+20 for division.
9. The code "5/6 6 6 6" represents an incomplete operation since it leaves the 'S516 in state 1 rather than in state 0, 8, or 10.

Figure 1. 'S516 Instruction Set (Partial List)

SUMMARY OF SIGNALS/PINS	
B_{15}-B_0	Bidirectional data bus inputs/outputs
I_2-I_0	Instruction (sequential control) input
CK	Clock pulse input
$\overline{GO}$	Chip activation input
OVR	Arithmetic overflow output

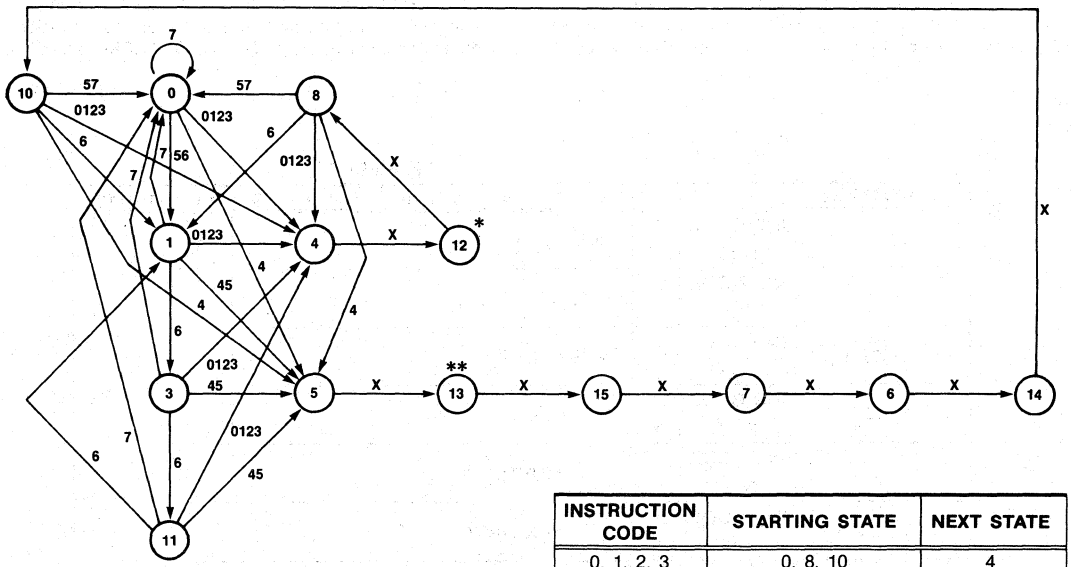
Description (continued)

The 'S516 device uses standard low-power Schottky technology, requires a single +5V power supply, and is fully TTL compatible. Bus inputs require at most 250 μA input current, and control and clock inputs require at most 1 mA input current. Bus outputs are three-state, and are capable of sinking 8 mA at the low logic level. The 'S516 is available in both commercial-temperature and military-temperature ranges, in a 600-mil 24-pin dual-in-line ceramic package.

Device Operation

The 'S516 contains four 16-bit working registers. Y is the multiplier register; X is the multiplicand and divisor register; W is the least-significant half of a double-length accumulator, and holds the least-significant half of the product after a multiplication operation, or the remainder after a division operation; and Z is the most-significant half of this same accumulator. In addition to these registers, there is a high-speed arithmetic unit which performs addition, subtraction, and shifting steps in order to accomplish the various arithmetic operations; a loading sequencer; and a PLA control network.

Operands are loaded into the working registers in time sequence at each clock period, under the control of this sequencer. The chip-activation signal $\overline{GO}$ must be LOW in order to begin the loading process and continue to the next step in the loading operation. If $\overline{GO}$ is continually held HIGH, the 'S516 remains in a wait state with its outputs held in their high-impedance states, so that the other devices attached to the bus may drive it. In this condition, the 'S516 does not respond to any codes on its instruction inputs; in effect, it does not "wake up" until $\overline{GO}$ goes LOW. Also, $\overline{GO}$ may change only when the clock input CK is HIGH. After all of the operands are loaded, the 'S516 jumps to the multiply routine, or to the divide routine, and performs the required operations as indicated in Figure 1. After 9 clock periods for a simple multiply or 21 clock periods for a simple divide, for example, the result is placed on the bus in time sequence.



INSTRUCTION CODE	STARTING STATE	NEXT STATE
0, 1, 2, 3	0, 8, 10	4
4	0, 8, 10	5
5	0	1
5, 7	8, 10	0
6	0, 8, 10	1
7	0, 8, 10	0

*Loop 7 times for multiplication.
**Loop 14 times for fractional division, or 15 times for integer division.

KEY:

The numbers inside the circles indicate the *state* of the 'S516 multiplier/divider. These states are represented by a four-bit state counter, where A is the least-significant bit of this state counter and D is the most-significant bit. (These four bits are not available externally on the 'S516.)

The next state of the 'S516 is a function of the present state and the instruction lines. For example if the 'S516 is at state 0 and the instruction is 0, 1, 2, or 3, then the next state is state 4 (multiply instruction); if the instruction is 4, the next state is state 5 (divide instruction); and so forth. The instructions which take the 'S516 from one state to another are indicated by the numbers written next to the state-transition path lines. "0123," for instance, implies that *any* of instructions 0, 1, 2, or 3 will take the 'S516 along the path marked "0123."

"X" next to a path implies that the path will be followed regardless of the value of the instruction inputs at that time. In other words, for the purpose of state transitions, X means "don't care." There are cases, however, where the particular instruction used may affect when the contents of the registers are available on the bus — see Figures 9 and 10 for contrasting examples of how this effect operates.

Figure 2. Transition Diagram for the 'S516 Multiplier/Divider

Three instruction inputs I_2, I_1, I_0, which may change only when the clock input CK is HIGH, select the required function and drive the sequencer from state to state. Thus, the action of the multiplier/divider at any clock period is a function of the machine state and the state of the control inputs. Figure 2 shows the multiply/divide state table, and all possible operations. After a Read or Round operation, the machine is driven back to state 0, and a new sequence of arithmetic operations is assumed. If a chain operation is being performed, such as accumulation of products, state 0 is bypassed, and loading of an operand or jumping to the next arithmetic operation occurs at the end of the previous arithmetic operation — at state 8 for a multiplication instruction, or at state 10 for a division instruction.

Register X is a dual-rank register, which allows the loading of an operand X during the multiplication or division process. If the machine enters the loading sequence and a new X operand has not been loaded, then the machine proceeds with the previously-loaded X, denoted in this text as "X1." This loading-while-processing capability allows a cycle to be saved during "chained" calculations, and also allows multiplication and division by a constant. (See Figure 13). (continued next page)

Figures 3 and 4 show the codes and durations for the 41 different possible arithmetic operations. These operations can be concatenated in strings to perform complicated 2s-complement arithmetic operations at high-speed. Rounding and reading of results can be performed after any operation. Figure 5 is a block diagram of the 'S516 16x16 Multiplier/Divider.

(continued page after next)

		TIME-SLOT											
OPERATION		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
X1 · Y	INS CODE	0	MULTIPLY										
	BUS	Y											
-X1 · Y	INS CODE	1	MULTIPLY										
	BUS	Y											
X1 · Y + K_Z, K_W	INS CODE	2	MULTIPLY										
	BUS	Y											
-X1 · Y + K_Z, K_W	INS CODE	3	MULTIPLY										
	BUS	Y											
X · Y	INS CODE	5/6	0	MULTIPLY									
	BUS	X	Y										
-X · Y	INS CODE	5/6	1	MULTIPLY									
	BUS	X	Y										
X · Y + K_Z, K_W	INS CODE	5/6	2	MULTIPLY									
	BUS	X	Y										
-X · Y + K_Z, K_W	INS CODE	5/6	3	MULTIPLY									
	BUS	X	Y										
X · Y + Z	INS CODE	5/6	6	0	MULTIPLY								
	BUS	X	Z	Y									
-X · Y + Z	INS CODE	5/6	6	1	MULTIPLY								
	BUS	X	Z	Y									
X · Y + $K_Z \cdot 2^{-16}$	INS CODE	5/6	6	2	MULTIPLY								
	BUS	X	—	Y									
-X · Y + $K_Z \cdot 2^{-16}$	INS CODE	5/6	6	3	MULTIPLY								
	BUS	X	—	Y									
X · Y + Z, W	INS CODE	5/6	6	6	0	MULTIPLY							
	BUS	X	Z	W	Y								
-X · Y + Z, W	INS CODE	5/6	6	6	1	MULTIPLY							
	BUS	X	Z	W	Y								
X · Y + W_{sign}	INS CODE	5/6	6	6	2	MULTIPLY							
	BUS	X	—	W	Y								
-X · Y + W_{sign}	INS CODE	5/6	6	6	3	MULTIPLY							
	BUS	X	—	W	Y								

NOTES: 1) X1 is the previous contents of the first rank of the X register (either old X or a new X).
2) $K_Z \cdot 2^{-16}$ is a single-length signed number comprising the most-significant half of the previous double-length product and here gets added in at the least-significant end of the new result.
3) W_{sign} is a single-length signed number, with sign-extension as needed.
4) Fractional or integer arithmetic is specified by having the next-to-the-last operand loaded using a 5 or 6 instruction respectively. All rows beginning with "5/6" in effect represent *two* instructions. 5 does fractional arithmetic and 6 does integer arithmetic.

Figure 3. Multiplication Codes and Times for 16x16 Multiplication in the 'S516

OPERATION		TIME-SLOT 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
$K_Z, K_W/X_1$	INS CODE	4	DIVIDE																			1			
	BUS	—																							
K_W/X	INS CODE	5/6	4	DIVIDE																		1			
	BUS	X	—																						
K_Z/X	INS CODE	5/6	5	DIVIDE																		1			
	BUS	X	—																						
Z, W/X	INS CODE	5/6	6	4	DIVIDE																		1		
	BUS	X	Z	W																					
Z/X	INS CODE	5/6	6	5	DIVIDE																		1		
	BUS	X	Z	—																					
W/X	INS CODE	5/6	6	6	4	DIVIDE																			1
	BUS	X	—	W	—																				
W_{sign}/X	INS CODE	5/6	6	6	5	DIVIDE																			1
	BUS	X	0	W	—																				

NOTES: 1) X1 is the previous contents of the first rank of the X register (either old X or a new X).
2) Fractional division divides a 32-bit 2s-complement number in 1 clock period less than integer division.
3) W_{sign} is a single-length signed number, with sign-extension as needed.
4) Division operation W_{sign}/X requires that the Z register be initialized with all-zero contents at the time Z is loaded.
5) Fractional or integer arithmetic is specified by having the next-to-the-last operand loaded using a 5 or 6 instruction respectively. All rows beginning with "5/6" in effect represent *two* instructions, one of which does fractional arithmetic and one of which does integer arithmetic.

Figure 4. Division Codes and Times for 32/16 Division in 'S516

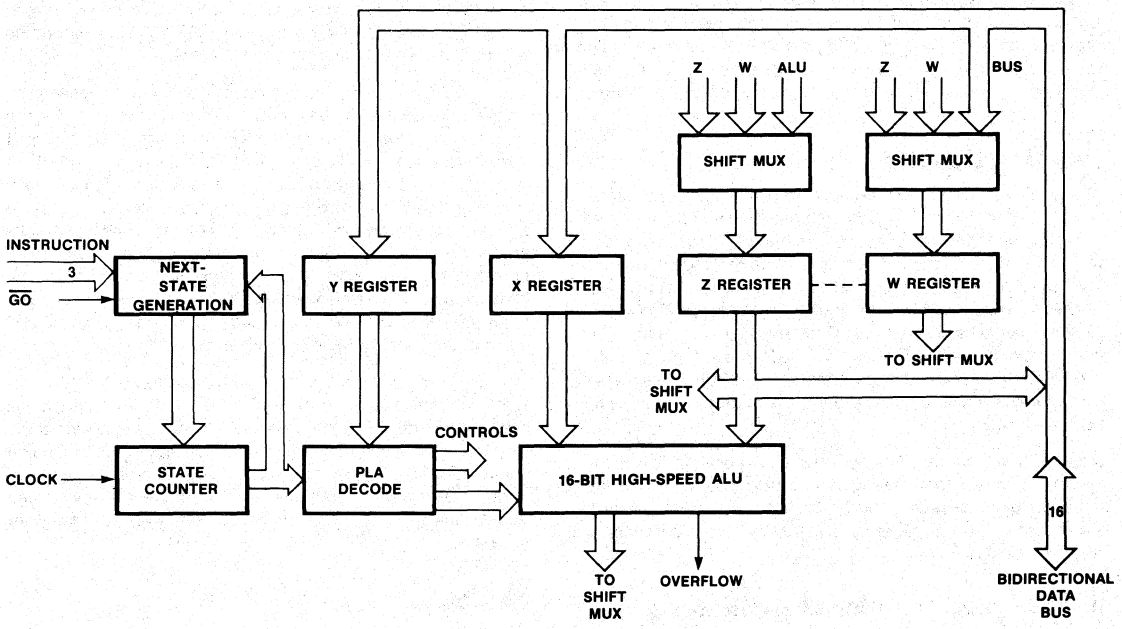


Figure 5. Internal Architecture of the 'S516

Initialization

The 'S516 has no direct master reset input. However, initialization of the 'S516 can easily be performed by continually presenting instruction code 7, which after a maximum of 21 clock periods forces the machine back to state 0.

Multiplication

The 'S516 provides 2s-complement 16-bit multiplication, and can also accumulate previously-generated double-length products. No time penalty is incurred for accumulation, since the machine accumulates while the multiplication operation is proceeding. In addition to accumulation, the device can add into a product either a single-length or a double-length number. It can also use a previously-loaded operand as a constant, so that constant multiplication and accumulation is possible.

One key feature is the ability to perform both positive multiplications and negative multiplications, again without any speed penalty. This feature allows complex-arithmetic multiplications to be programmed with very little overhead. Another important feature is the ability to work with either fractions or integers.

Division

The 'S516 also provides a range of division operations. A double-length number in Z,W is divided by X; the result Q is stored in Z, and the remainder R in W. Again all numbers are in the 2s-complement number representation, with the most significant bit of an operand (whether single-length or double-length) having a negative weight. In order to facilitate repeated division, with the multiple-length quotient always keeping the same sign, the remainder is always the same sign as the dividend. Fractional or integer operation is possible, and division and multiplication operations can be concatenated. For example, the operations (AxB)/C, (A + B)/C can easily be performed. The dividend can be any previously-generated result — product, quotient, or remainder; or it may be a double-length or single-length signed operand.

Reading Results

The result of an arithmetic operation, or of a string of operations, can be read onto the 16-bit bus if the machine is at the end of an operation or at the start of a new sequence. The read operation requires that the $\overline{GO}$ signal be held LOW so that the information is read out onto the bidirectional bus, when code 7 is specified. (See Figure 6.) Since there is a double-length accumulator Z,W, reading can take two cycles. First, register Z is read. After another clock has been received, if code 7 is still present, the least-significant half of the product from the W register is placed on the bus, or likewise the remainder if a division operation had been performed.

If the 'S516 is instructed to perform a read operation during the loading sequence, then the sequence is broken and the machine is forced back to state 0 ready to start the sequence again. Control read operations at state 0 just swap the contents of register Z and W.

Integer and Fractional Arithmetic

The 'S516 can work with either fractional or integer number representations. When working with integers, all numbers are scaled from the least-significant end, and the least-significant bit is assumed to have a weight of 2^0. For integer multiplication, accumulation, and division, all numbers are scaled from this least-significant weight, and results are correct if interpreted in this manner. The double-length register Z,W can therefore hold numbers in the range -2^{31} to $+2^{31}-1$; the operands X and Y, and single-length results, are in the range -2^{15} to $+2^{15}-1$.

When working with fractions, the machine automatically performs scaling so that input operands and results have a consistent format. All numbers in the fractional representation are scaled from the most significant end, which has a weight of -2^0 *(negative)*. The binary point is one place to the right of this most-significant bit, so that the next bit has a weight of 2^{-1}. The double-length register Z,W therefore holds numbers in the range -1 to $+1-2^{-31}$ and the operands X and Y and single-length results are in the range -1 to $+1-2^{15}$. Since automatic scaling occurs, the product of two numbers always has the least-significant bit as a 0, unless an accumulation is performed with the least-significant bit being a 1.

During a chain operation with the partial results not being read onto the bus, the 'S516 will stay in either the fractional or integer mode. At the start of a sequence of operations, fractional or integer operation is designated by loading operands using instruction code 5 or instruction code 6 respectively.

Mixed fractional and integer arithmetic is also possible, by redefining the weight of the least-significant or most-significant bits. However, care must be exercised, due to the automatic scaling feature, when fractional arithmetic is programmed.

Rounding

Rounding can be performed on the result of a multiplication or division. Generally rounding would only be called out during fractional operation, but nothing in the 'S516 precludes forming a rounded result during integer arithmetic.

Rounding for multiplication provides the best single-length most-significant half of the product. Rounding occurs at the end of a multiplication, and is performed instead of a Load or Read operation when a code 5 is specified, instead of a code 7, to get from state 8 or state 10 back to state 0. (See Figure 2; also, note that this mode of operation precludes "stealing" a cycle according to the method illustrated in Figure 9.) The 'S516 looks at the most-significant bit of the least-significant half of the product W_{15}, and adds 1 to the most-significant half of the product at the least-significant end if W_{15} is a 1. After the operation, the 'S516 is in state 0, so that the rounded product can be read, and the W register is cleared.

Rounding for division is performed by forcing the least-significant bit of the quotient in Z to a 1 unless the division is exact (remainder is zero). This method of rounding causes a slightly higher variance in the result than having an additional iterative division operation, but is considerably easier to perform. Again, after rounding the 'S516 goes to state 0, so that a read operation can be performed, and the W register is cleared.

Overflow

The 'S516 has an overflow output OVR which is cleared prior to each operation, and is set during an operation if the product or quotient goes outside the normally-accepted range.

For multiplication, overflow can only occur if the most negative number in the operand range is used: $(-1)x(-1)=+1$, which cannot be held in the 'S516's internal registers. Overflow can more easily occur during either positive or negative accumulation of products. For fractional arithmetic, if the product or accumulation goes outside the range of -1 to $+1-2^{-31}$, then the overflow flipflop will be set.

The overflow flip-flop is enabled in state 8 for the multiply operation or in state 10 for a divide operation. It only gets reset when a transition to state 0 from states 0, 3, 8, 10 and 11, when instruction 7 is being presented to the 'S516.

Overflow may also occur during division if the quotient goes outside the generally-accepted number range of -1 to $+1-2^{-15}$ during fractional operation. This would occur if the divisor is less than the dividend, or equal to the dividend if a positive quotient is being generated. For integer arithmetic the numbers must be scaled by 2^{15}.

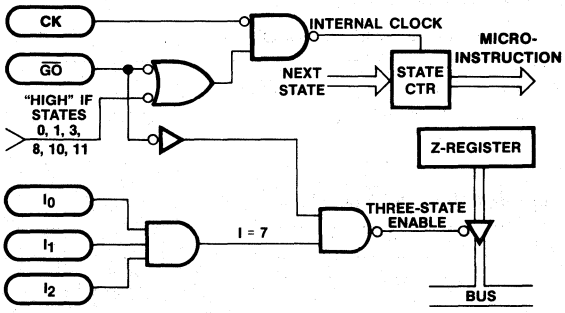


Figure 6. 'S516 Internal Circuitry of "$\overline{GO}$" Line and Three-State-Enable

During the states 0, 1, 3, 8, 10 and 11 if the "GO" line ($\overline{GO}$) is held at logic HIGH then the machine will be in a wait state until $\overline{GO}$ goes to logic LOW.

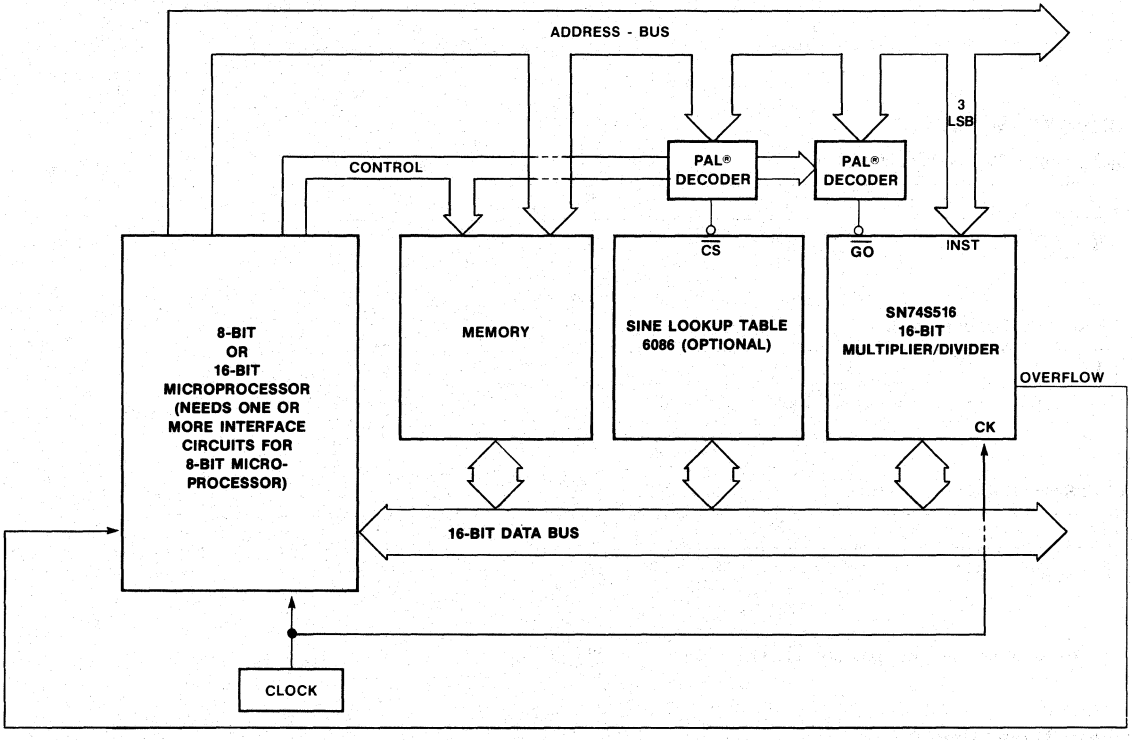


Figure 7. Interfacing the 'S516 to a Microprocessor

Figure 7 shows the block diagram of a microprocessor system with its arithmetic capabilities enhanced by the use of a 'S516 16x16 multiplier/divider. The relatively small number of instruction lines (only 3) of the 'S516 provides a unique way to control the multiplier/divider. As may be seen from Figure 7, these three instruction lines are assigned to the three least-significant bits (LSBs) of the address bus, while the remaining address bits are decoded by a Programmable Array Logic (PAL®) circuit to determine when the multiplier/divider is selected. For example, suppose the 'S516 is assigned address 100; then any address in the range of 100-107 will enable the 'S516 (i.e., the $\overline{GO}$ line is LOW). Thus, if the address is 100 the 'S516 instruction is 0; if the address is 106 the 'S516 instruction is 6; and so forth.

Data Formats

Fractional Multiply

X_i, Y_1 - Input, Multiplicand, Multipler

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Sign	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	2^{-14}	2^{-15}

Z_i - MS Half Output Product

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Sign	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	2^{-14}	2^{-15}

W_i - LS Half Output Product*

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
2^{-16}	2^{-17}	2^{-18}	2^{-19}	2^{-20}	2^{-21}	2^{-22}	2^{-23}	2^{-24}	2^{-25}	2^{-26}	2^{-27}	2^{-28}	2^{-29}	2^{-30}	"0"

* The least significant bit of W_i is always a binary 0 due to normalization. Note that -1 x -1 yields an overflow in fractional multiply.

Integer Multiply

X_i, Y_1 - Input, Multiplicand, Multiplier

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Sign	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

Z_i - MS Half Output Product

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Sign	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}

W_i - LS Half Output Product**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

** The least significant bit of W_i is a valid data bit. Note that $2^{-15} \times 2^{-15}$ yields $+2^{30}$ which can be represented in the output bits without overflowing.

Fractional Divide

Z_i - Input Dividend

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Sign	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	2^{-14}	2^{-15}

X - Input Divisor

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Sign	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	2^{-14}	2^{-15}

Z_i - Output Quotient

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Sign	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	2^{-14}	2^{-15}

W - Output Partial Remainder †

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Sign	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	2^{-14}	2^{-15}

† Note that the partial remainder $R = 2^{-15}$ (W)

Integer Divide Example (Z, W)/X

Z_i - MSB Input Dividend

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Sign	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}

W_i - LSB Input Dividend

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

X - Input Divisor

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Sign	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

Z_i - Output Quotient

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Sign	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

W_i - Output Partial Remainder

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Sign	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

Absolute Maximum Ratings

Supply voltage V_{CC} ..... 7.0 V
Input voltage ..... 7.0 V
Off-state output voltage ..... 5.5 V
Storage temperature ..... −65° to +150°C

Operating Conditions

SYMBOL	PARAMETERS	FIGURE	COMMERCIAL MIN	COMMERCIAL TYP	COMMERCIAL MAX	UNIT
V_{CC}	Supply voltage		4.75	5	5.25	V
T_A	Operating free-air temperature		0		75	°C
f_{MAX}	Clock frequency	8	6			MHz
t_{CWP}	Positive clock pulse width	8	70			ns
t_{CWN}	Negative clock pulse width	8	50			ns
t_{BS}	Bus setup time for inputting data*	8	50			ns
t_{BH}	Bus hold time for inputting data*	8	35			ns
t_{INSS}	Instruction, GO setup time	8	10			ns
t_{INSH}	Instruction, GO hold time	8	30			ns

* During operations when the bus is being used to input data.

Electrical Characteristics Over Operating Conditions

SYMBOL	PARAMETER	TEST CONDITIONS		MIN	TYP	MAX	UNIT
V_{IL}	Low-level input voltage					0.8	V
V_{IH}	High-level input voltage			2			V
V_{IC}	Input clamp voltage	V_{CC} = MIN I_I = −18mA				−1.5	V
I_{IL}	Low-level input current	V_{CC} = MAX V_I = 0.5V	B_{15}–B_0			−250	μA
			All other inputs			−1	mA
I_{IH}	High-level input current	V_{CC} = MAX V_I = 2.4V				250	μA
I_I	Maximum input current	V_{CC} = MAX V_I = 5.5V				1	mA
V_{OL}	Low-level output voltage	V_{CC} = MIN I_{OL} = 8mA			0.3	0.5	V
V_{OH}	High-level output voltage	V_{CC} = MIN I_{OH} = −2mA		2.4			V
I_{OS}	Output short-circuit current*	V_{CC} = MAX V_O = 0V		−10		−90	mA
I_{CC}	Supply current	V_{CC} = MAX			370	450†	mA

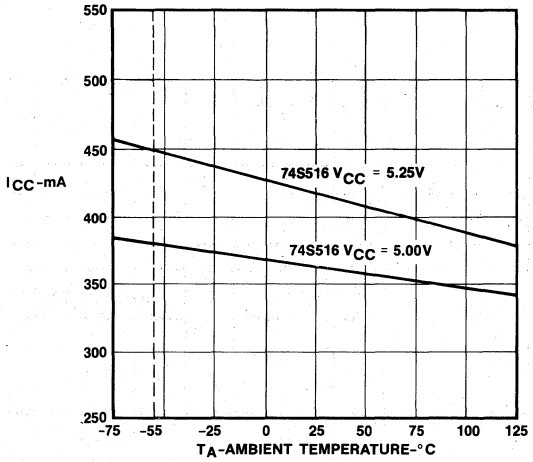
* Not more than one output should be shorted at a time, and the duration of the short-circuit should not exceed one second.
† At cold temperatures see the "I_{CC} vs Temperature" curves on the next page for more complete information. The typical values shown here are at 5.0 V.

Switching Characterictics Over Operating Conditions

SYMBOL	PARAMETER		FIGURE	COMMERCIAL MIN	COMMERCIAL TYP	COMMERCIAL MAX	UNIT
t_{BO}	Bus output delay from CK for outputting data;* C_L = 30 pF		8		70	95	ns
t_{PXZ}	Output disable delay	FROM I_2–I_0 to bus			30	65	ns
		From GO to bus			20	40	
t_{PZX}	Output enable delay; C_L = 30 pF	FROM I_2–I_0 to bus			55	80	ns
		From GO to bus			25	45	
t_{OVR}	Overflow output delay from CK; C_L = 30 pF		8		60	95	ns

* During operations when the bus is being used to output data.

I_{CC} vs. Temperature

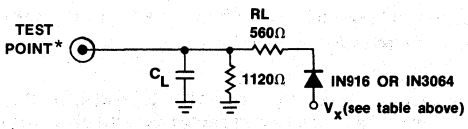


Test Waveforms

TEST	V_X*		OUTPUT WAVEFORM — MEAS. LEVEL
All t_{PD}	5.0V		V_{OH}, V_{OL}, 1.5V
t_{PXZ}	t_{PHZ} 0.0V	t_{PLZ} 5.0V	V_{OH}, 0.5V, 0.5V, V_{OL}, 2.8V, 0.0V
t_{PZX}	t_{PZH} 0.0V	t_{PZL} 5.0V	2.8V, 0.0V, 1.5V, V_{OH}, V_{OL}

*At diode; see "Test Circuit" figure below.

Test Load



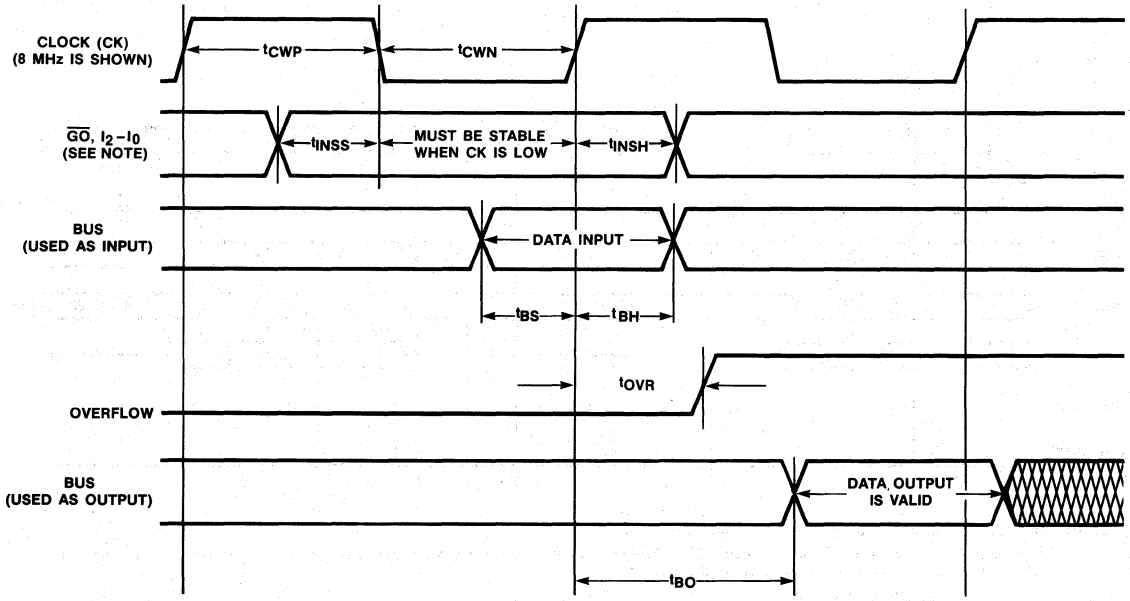
* The "TEST POINT" is driven by the output under test, and observed by instrumentation.

AC Test Conditions

Inputs 0 V_{LOW}, 3 V_{HIGH}. Rise and fall time 1-3 ns from 1 V to 2 V. Measurements are made from 1.5 V_{IN} to 1.5 V_{OUT}, except that t_{PXZ} is measured by a delta in the outputs of 0.5 V from V_{OL} or V_{OH} respectively.

Timing

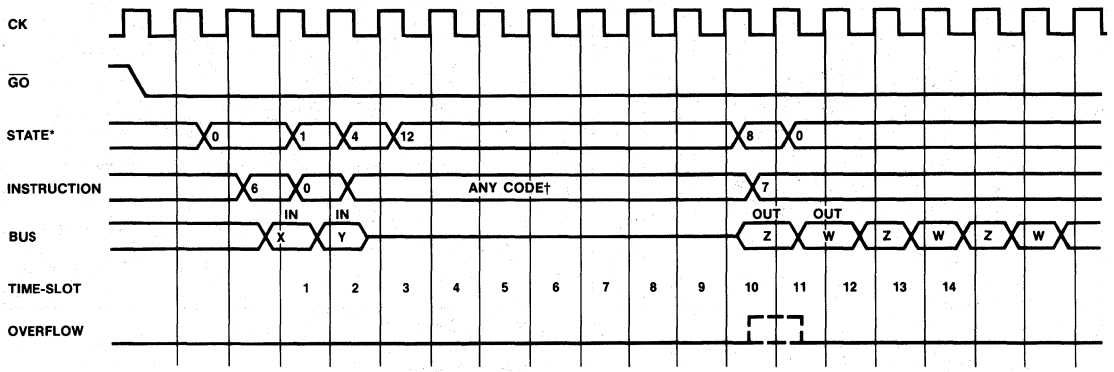
Timing waveforms are shown in Figure 8. Specific instruction timing examples are shown in Figures 9 through 13.



NOTE: $\overline{GO}$ and I_2-I_0 can change only when CK is high.

Figure 8. Timing Diagram of the 'S516

10

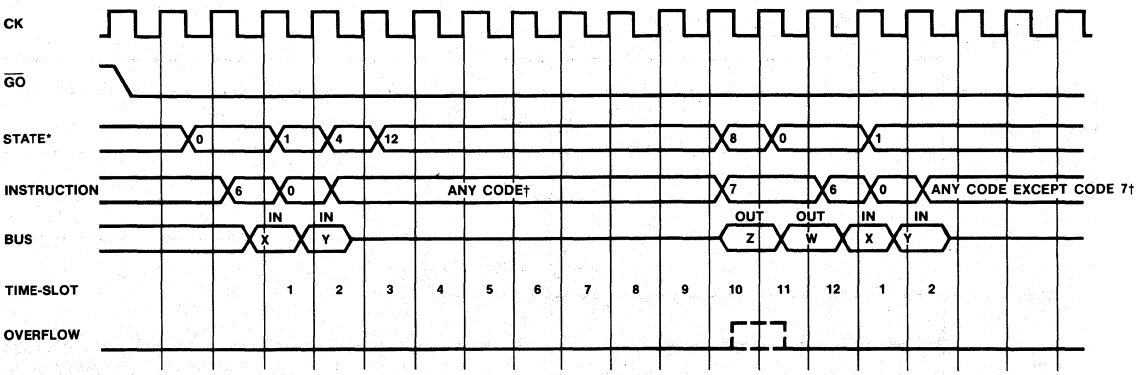


NOTES: Register Z is read at the same time that the overflow signal (if present) is set. If the instruction remains at code 7 after time-slot 11, the contents of registers Z and W are swapped each cycle.

†"Any code" means any of code 0 through code 7. However, code 6 will load a new value of X, and code 7 will cause the 'S516 to attempt to drive the data bus.

*Not available externally on the 'S516.

Figure 9. Instruction Timing Example No. 1: Load X, Load Y, Multiply, Read Z, Read W. By Presenting Code 7 on the Instruction Lines During the Last Multiply Cycle (State 8), the Results May Be Read During Time-Slots 10 and 11

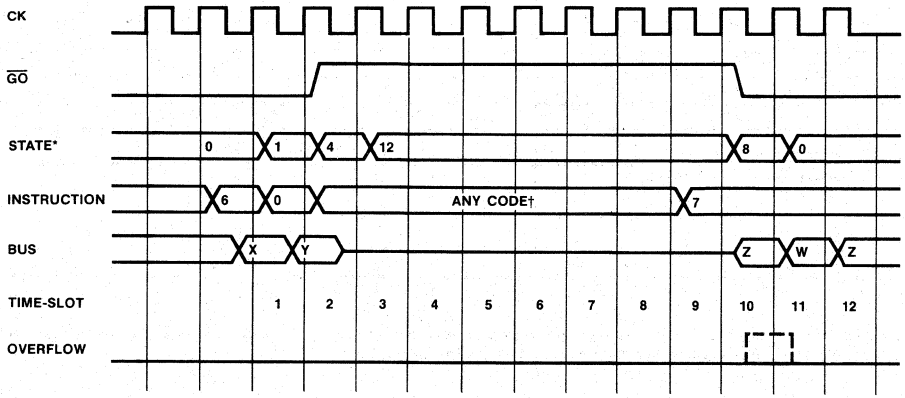


NOTES: The instruction lines may be changed only when CK is high.

†"Any code" means any of code 0 through code 7. Code 6 may be used here since a new X explicitly gets loaded for the next multiply operation. However, code 7 will cause the 'S516 to attempt to drive the data bus.

*Not available externally on the 'S516.

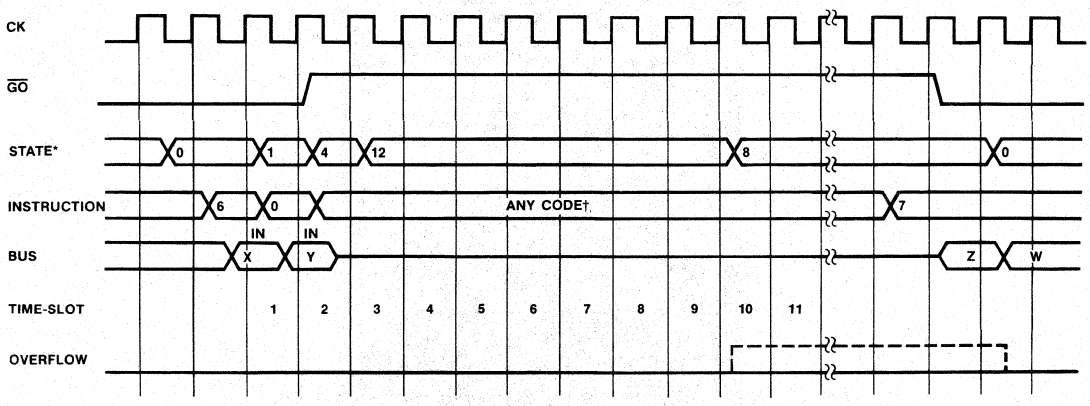
Figure 10. Instruction Timing Example No. 2: Repeat: "Load X, Load Y, Multiply, Read Z, Read W"



NOTES: Code 7 is given in time-slot 9, but has no effect until time-slot 10 since $\overline{GO}$ is HIGH. After $\overline{GO}$ goes LOW in time-slot 10, Z may be read.
†"Any code" means any of code 0 through code 7.
*Not available externally on the 'S516.

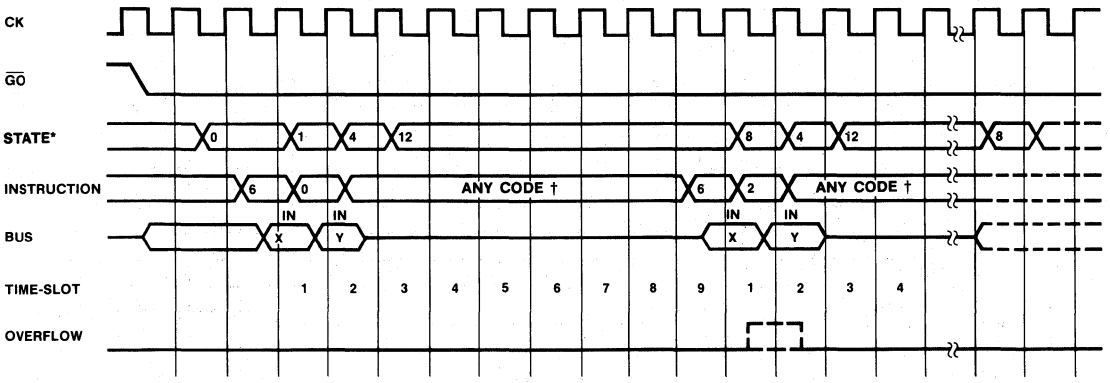
Figure 11. Instruction Timing Example No. 3: Load X, Load Y, Multiply, Read Z, Read W. This Timing Diagram Corresponds to Table 1. Only After Eight Clock Pulses of the Operation Cycle, the Result Is Read — Z During Time-Slot 10 and W During Time-Slot 11

10



NOTES: †"Any code" means any of code 0 through code 7. Code 6 or code 7 may be used here; since $\overline{GO}$ is HIGH, no new X can be loaded, and the 'S516 cannot attempt to drive the data bus.
*Not available externally on the 'S516.

Figure 12. Instruction Timing Example No. 4: Load X, Load Y, Multiply, Wait, Read Z, Read W



NOTES: This sequence of operations is suitable for use when reading is to be done only at the very end of the operation sequence. The new X value is loaded during the time that the previous multiplication is being performed. See Programming Example #3 for $\sum_{i=1}^{N} X_i \cdot Y_i$

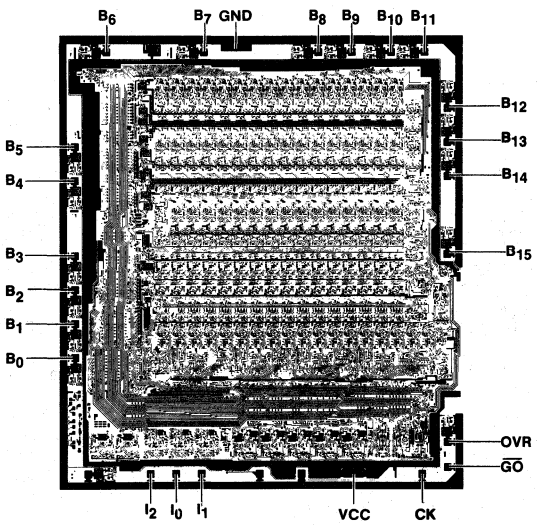
†"Any code" means any of code 0 through code 7. However, code 7 will cause the 'S516 to attempt to drive the data bus.

*Not available externally on the 'S516.

††Code 6 allows loading of a new X in State 12 and it takes the 'S516 State Counter to State 8. In State 8, Y is loaded via instruction 2 and the next multiply-accumulate cycle is initiated.

Figure 13. Instruction Timing Example No. 5: Sum of Products

Die Configuration



Die Size: 210x234 mil^2

Programming Examples

In the following examples assume that each line with a separate instruction corresponds to one clock pulse. Instruction codes are 0, 1, 2, 3, 4, 5, 6, 7 and x according to the usage explained in the key to Figure 2.

Programming Example 1

Calculating X · Y (A·B)

INST 6	X ← A
INST 0	Y ← B
INST X	MULT
INST X	MULT
INST X	MULT
INST X	MULT
INST X	MULT
INST X	MULT
INST X	MULT
INST 7	MULT AND READ Z = 16 MSB OF (A·B)
INST 7	READ W = 16 LSB OF (A·B)

Programming Example 2

Calculating X1 · Y (A·C)

X1 is a previous multiplier value. It was previously loaded (in example 1) with A.

INST 0	Y ← C
INST X	MULT
INST X	MULT
INST X	MULT
INST X	MULT
INST X	MULT
INST X	MULT
INST X	MULT
INST 7	MULT and READ Z = 16 MSB OF (A·C)
INST 7	READ W = 16 LSB OF (A·C)

Programming Example 3

Calculating $\sum_{i=1}^{N} X_i \cdot Y_i$ (A·B + C·D + E·F + . . .)

In this case we read only after N multiplications. A new X_{i+1} is loaded during the multiplication process for X_iY_i.

Assume N = 3.

The sequence of instructions and operations for calculating $\sum_{i=1}^{3} X_i \cdot Y_i$ is: (A·B + C·D + E·F)

N = 1	INST 6	X ← A
	INST 0	Y ← B
	INST X	MULT
	INST X	MULT
	INST X	MULT
	INST X	MULT
	INST X	MULT
	INST X	MULT
	INST X	MULT
		Perform A·B
N = 2	INST 6	MULT and LOAD X ← C Z ← 16 MSB of (A·B) W ← 16 LSB of (A·B)
	INST 2	Y ← D
	INST X	MULT
	INST X	MULT
	INST X	MULT
	INST X	MULT
	INST X	MULT
	INST X	MULT
	INST X	MULT
		Perform C·D + (K_Z, K_W)
N = 3	INST 6	MULT and LOAD X ← E Z ← 16 MSB of (C·D + A·B) W ← 16 LSB of (C·D + A·B)
	INST 2	Y ← F
	INST X	MULT
	INST X	MULT
	INST X	MULT
	INST X	MULT
	INST X	MULT
	INST X	MULT
	INST X	MULT
		Perform E·F + (K_Z, K_W)
READ Z	INST 7	MULT and READ Z = 16 MSB of (E·F + C·D + A·B)
READ W	INST 7	READ W = 16 LSB of (E·F + C·D + A·B)

10

Programming Example 4

Multiplication plus a constant (A·B + Constant)
Assume that the constant is a 32-bit 2s-complement number.

INST 6	X ← A	
INST 6	Z ← C	LOAD 16 MSB of constant
INST 6	W ← D	LOAD 16 LSB of constant
INST 0	Y ← B	
INST X	MULT	} Perform A·B + (Z, W)
INST X	MULT	
INST X	MULT	
INST X	MULT	
INST X	MULT	
INST X	MULT	
INST X	MULT	
INST 7	MULT and READ Z = 16 MSB of (A·B + (C, D))	
INST 7	READ W = 16 LSB of (A·B + (C, D))	

Programming Example 5

Dividing a 32-bit number by a 16-bit number ((B, C)/A)

INST 6	X ← A	
INST 6	Z ← B	
INST 4	W ← C	
INST X		} Perform Division $\frac{(Z, W)}{X}$
INST X		
INST X		
INST X		
INST X		
INST X		
INST X		
INST X		
INST X		
INST X		
INST X		
INST X		
INST X		
INST X		
INST X		
INST X		
INST X		
INST 7	DIVIDE and READ the quotient Z = $\frac{(B, C)}{A}$	
INST 7	READ the remainder W of $\frac{(B, C)}{A}$	

16x16 Flow-Thru™ Multiplier Slice 54/74S556

Features/Benefits

- **Twos-complement, unsigned, or mixed operands**
- **Full 32-bit product immediately available on each cycle**
- **High-speed 16x16 parallel multiplier**
- **Latched or transparent inputs/outputs**
- **Three-state output controls, independent for each half of the product**
- **Single +5 V supply (via multiple pins)**
- **Available in 84-terminal Leadless-Chip Carrier and 88-Pin-Grid-Array packages**

Description

The 'S556 is a high-speed 16x16 combinatorial multiplier which can multiply two 16-bit unsigned or signed twos-complement numbers on every cycle. Each operand X and Y has an associated mode-control line, XM and YM respectively. When a mode-control line is at a LOW logic level, the operand is treated as an unsigned 16-bit number; when the mode-control line is at a HIGH logic level, the operand is treated as a 16-bit signed twos-complement number. Additional inputs RS and RU allow the addition of a bit into the multiplier array at the appropriate bit positions for rounding. The entire 32-bit double-length product is available at the outputs at one time.

Ordering Information

PART NUMBER	PACKAGE	TEMPERATURE
54S556	P88, L84*	Military
74S556	P88, L84*	Commercial

P88 is an 88-Pin-Grid-Array Package.

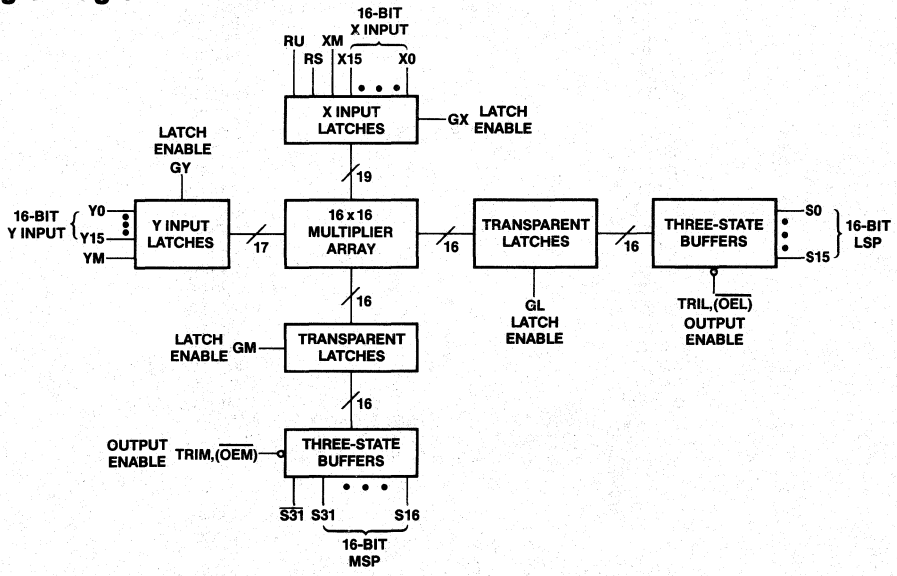
L84 is an 84-terminal Leadless-Chip Carrier Package.

* The 84-terminal leadless chip carrier, L84, and its socket, L84-2, are in development; contact the factory for further details.

The most-significant product bit, S31, is available in both true and complemented form to simplify longer-wordlength multiplications. The product outputs are three-state, controlled by assertive-low enables. The MSP outputs are controlled by the TRIM ($\overline{OEM}$) control input, while the LSP outputs are controlled by the TRIL ($\overline{OEL}$) control input. This allows one or more multipliers to be connected to a parallel bus or to be used in a pipelined system.

All inputs and outputs have transparent latches. The latches become transparent when the input to the corresponding gate control line GX, GY, GM, GL is HIGH. If latches are not required, these control inputs may be tied HIGH, leaving the multiplier fully transparent for combinatorial cascading. The device uses a single +5 V power supply, and is available both in an 84-terminal leadless chip carrier (LCC) package and in an 88-pin-grid-array package.

'S556 Logic Diagram



10

2175 Mission College Blvd. Santa Clara, CA 95054-1592 Tel: (408) 970-9700 TWX: 910-338-2374
TWX: 910-338-2376
Monolithic Memories MMI

SUMMARY OF SIGNALS/PINS	
X_{15-0}	Multiplicand 16-bit data inputs
Y_{15-0}	Multiplier 16-bit data inputs
XM, YM	Mode-control inputs for each data word; LOW for unsigned data and HIGH for twos-complement data
S_{31-0}	Product 32-bit output
$\overline{S}_{31}$	Inverted MS product bit (for expansion)
RS, RU	Rounding inputs for signed and unsigned data, respectively
GX	Gate control for X_i, RS, RU
GY	Gate control for Y_i
GL	Gate control for least-significant half of product
GM	Gate control for most-significant half of product
TRIL / $\overline{OEL}$	Three-state control for least-significant half of product
TRIM / $\overline{OEM}$	Three-state control for most-significant half of products

Rounding Inputs

INPUTS		ADDS		USUALLY USED WITH	
RU	RS	2^{15}	2^{14}	XM	YM
L	L	NO	NO	X	X
L	H	NO	YES	H†	H†
H	L	YES	NO	L	L
H	H	YES	YES	*	*

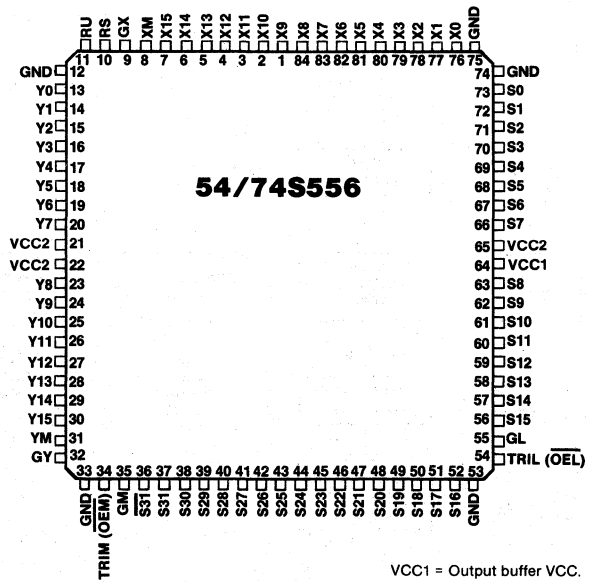
† In mixed mode, one of these could be low but not both.

* Usually a nonsense operation.

Mode-Control Inputs

OPERATING MODE	INPUT DATA		MODE-CONTROL INPUTS	
	X_{15-0}	Y_{15-0}	XM	YM
Unsigned	Unsigned	Unsigned	L	L
Mixed	Unsigned	Twos-Comp.	L	H
	Twos-Comp.	Unsigned	H	L
Signed	Twos-Comp.	Twos-Comp.	H	H

84-Terminal Leadless Chip Carrier Pinout



VCC1 = Output buffer VCC.
VCC2 = Logic VCC.

All V_{CC} and GND pins must be connected to the respective V_{CC} and GND connections on the board and should not be used for daisychaining through the IC.

Operating Conditions

SYMBOL	PARAMETER		FIGURE	MILITARY MIN	MILITARY TYP	MILITARY MAX	COMMERCIAL MIN	COMMERCIAL TYP	COMMERCIAL MAX	UNIT
V_{CC}	Supply voltage			4.5	5	5.5	4.75	5	5.25	V
T_A	Operating free-air temperature			-55		*125	0		75	°C
t_{S1}	Setup time $(X_i, R_i)/Y_i$ to GX/GY		2a, 2b	12			10			ns
t_{S2}	Setup time X_i, Y_i, R_i to GM, GL	t_{S2L}	3a, 3b	65			60			ns
		t_{S2M}		82			74			
t_{S3}	Setup time GX, GY to GL, GM	t_{S3L}	4a, 4b, 4c, 4d, 4e, 4f	65			60			ns
		t_{S3M}		85			75			
t_{H1}	Hold time $(X_i, R_i)/Y_i$ to GX/GY		2a, 2b	8			8			ns
t_{H2}	Hold time X_i, Y_i, R_i to GM, GL	t_{H2L}, t_{H2M}	3a, 3b	3			3			ns
t_{H3}	Hold time GX, GY to GM, GL	t_{H3L}, t_{H3M}	4a, 4b, 4c, 4d, 4e, 4f	0			0			ns
t_w	Latch enable pulse width		6	14			12			ns

* Indicates case temperature.

Electrical Characteristics Over Operating Conditions

SYMBOL	PARAMETER	TEST CONDITIONS		MIN	TYP†	MAX	UNIT
V_{IL}	Low-level input voltage**					0.8	V
V_{IH}	High-level input voltage**			2			V
V_{IC}	Input clamp voltage	V_{CC} = MIN	I_I = -18 mA			-1.5	V
I_{IL}	Low-level input current	V_{CC} = MAX	V_I = 0.4 V			-0.4	mA
I_{IH}	High-level input current	V_{CC} = MAX	V_I = 2.4 V			75	μA
I_I	Maximum input current	V_{CC} = MAX	V_I = 5.5 V			1	mA
V_{OL}	Low-level output voltage	V_{CC} = MIN	I_{OL} = 8 mA			0.5	V
V_{OH}	High-level output voltage	V_{CC} = MIN	I_{OH} = -2 mA	2.4			V
I_{OZL}	Off-state output current	V_{CC} = MAX	V_O = 0.5 V			-100	μA
I_{OZH}			V_O = 2.4 V			100	μA
I_{OS}	Output short-circuit current*	V_{CC} = MAX	V_O = 0 V	-20		-90	mA
I_{CC}	Supply current	V_{CC} = MAX	Commercial 74S556		600	800	mA
			Military 54S556		600	900	mA
I_{CC}	Supply current at hot temperature limit	V_{CC} = 5.25 V	T_A = 75°C			700	mA
		V_{CC} = 5.5 V	T_C = 125°C			800	mA

† Typical at 5.0 V and 25°C T_A.

* Not more than one output should be shorted at a time and the duration of the short-circuit should not exceed one second.

** These are absolute voltages with respect to the ground pins and include all overshoots due to system and/or tester noise. Do not attempt to test these values without suitable equipment.

10

Switching Characteristics Over Operating Conditions

SYMBOL	PARAMETER		TEST CONDITIONS	54S556 MILITARY			74S556 COMMERCIAL			UNIT
				MIN	TYP	MAX	MIN	TYP	MAX	
t_{DTL}	Transparent Multiply GX, GY, GM, GL = H	X_i,Y_i,R_i to S_{15-0} Figs. 1, 2c, 3b, 4c, 4f	CL = 30 pF RL = 560Ω See figure 7		84			50	76	ns
t_{DTM}		X_i,Y_i,R_i to $S_{31},\overline{S}_{31-16}$ Figs. 1, 2c, 3b, 4c, 4f			100			60	90	ns
t_{D1L}	Transparent Output Multiply GM, GL = H	GX, GY to S_{15-0} Figs. 2a, 2b, 4d, 4e			88				80	ns
t_{D1M}		GX,GY, to $S_{31},\overline{S}_{31-16}$ Figs. 2a, 2b, 4d, 4e			102				92	ns
t_{D2}	Transparent Input Multiply GX, GY = H	GM, GL to S_i Figs. 3a, 4a, 4b			40				35	ns
t_{PXZ}	Three-State Disable Timing	TRIL ($\overline{OEL}$), TRIM ($\overline{OEM}$) to S_i Fig. 5			40				30	ns
t_{PZX}	Three-State Enable Timing	TRIL ($\overline{OEL}$), TRIM ($\overline{OEM}$) to S_i Fig. 5			40				30	ns

Transparent Multiply — Flowthrough Operation

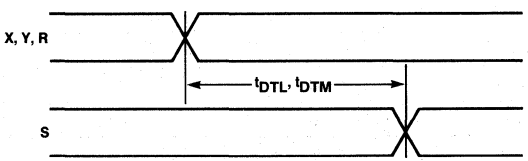
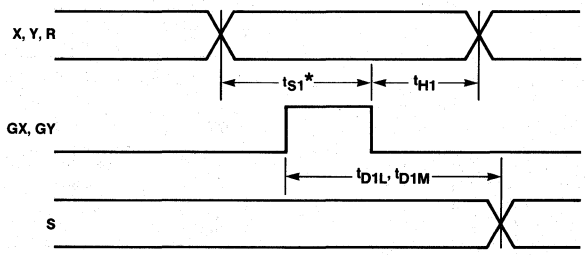


Figure 1.

The transparent multiply is a flowthrough operation of the 'S556. Both the input and output latches are made transparent by keeping GX, GY, GM, and GL at a HIGH level. The operands are presented to the X, Y, and R inputs; the results are available t_{DTL} and t_{DTM} later, for the least and most significant halves of the product respectively.

Transparent Output Multiply — Pipelined Input



* With this particular timing, set-up time t_{S1} will automatically be met.

Figure 2a.

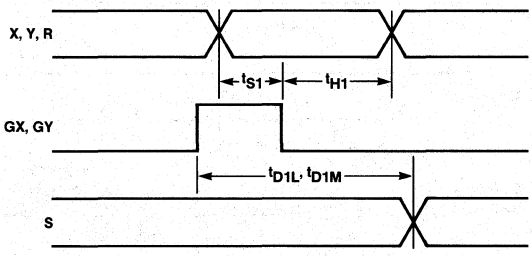


Figure 2b.

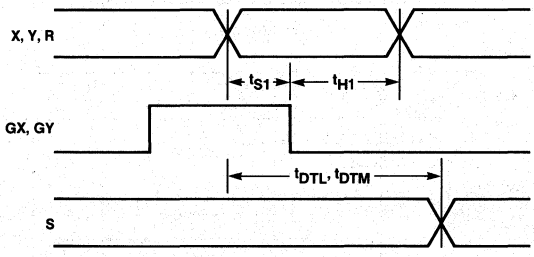


Figure 2c.

By tying the GL and GM lines HIGH, the 'S556 can perform transparent output (or pipelined input) multiplies. Data present is latched at the inputs using the GX and GY control signals. The time at which the result S is present at the outputs depends on when the rising edges of GX and GY occur. If the rising edges of GX and GY occur *after* the operand inputs change, then Figure 2a applies; the result will be available at the outputs t_{D1L} and t_{D1M}* after the rising edges of GX and GY. If the rising edges of GX and GY occur *less than* ($t_{W\ min}$ - $t_{S1\ min}$) *before* the operand inputs change, then Figure 2b applies; in this case the result will also be available at the outputs t_{D1L} and t_{D1M}* after the rising edges of GX and GY. However, if the rising edges of GX and GY occur *more than* ($t_{W\ min}$ - $t_{S1\ min}$) *before* the operand inputs change, then Figure 2c applies; the result will appear at the outputs t_{DTL} and t_{DTM}* after the operand inputs change.

* For the least and most significant halves of the product, respectively.

10

Transparent Input Multiply — Pipelined Output

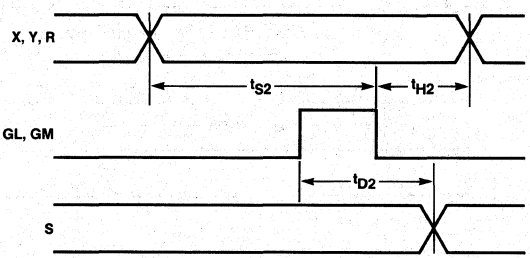


Figure 3a.

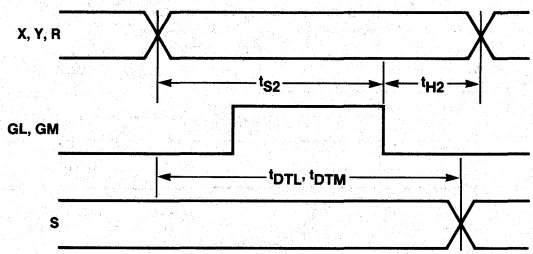


Figure 3b.

By tying the GX and GY lines HIGH, the 'S556 can perform transparent input (or pipelined output) multiplies. Data is presented at the inputs, and t_{S2} after X, Y and R change, the results can be latched. The time at which the result S is present at the outputs depends upon when the rising edges of GL and GM occur. If they occur *at or after* ($t_{S2\ min}$ - $t_{W\ min}$) from the inputs changing, then Figure 3a applies; the result appears at the outputs t_{D2} after the rising edges of GL and GM. If the rising edges of GL and GM occur *before* ($t_{S2\ min}$ - $t_{W\ min}$) from the inputs changing, then Figure 3b applies; the result appears at the outputs t_{DTL} and t_{DTM}* after the operand inputs change.

* For the least and most significant halves of the product, respectively.

Gated Multiply — Pipelined Input and Output

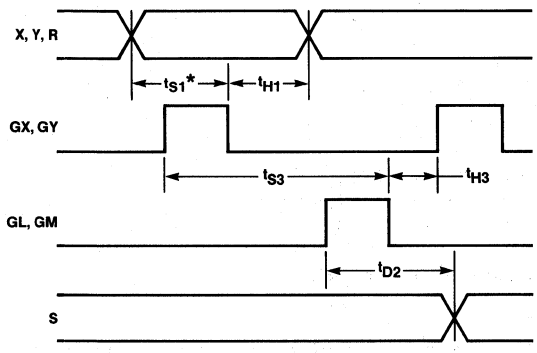


Figure 4a.

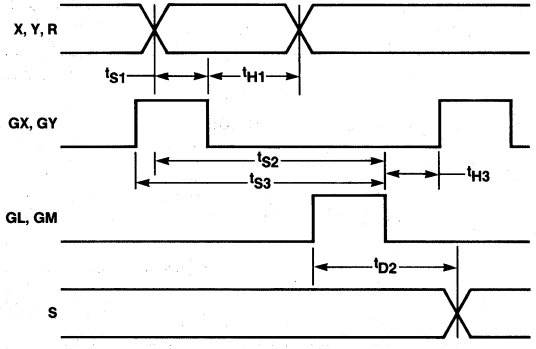


Figure 4b.

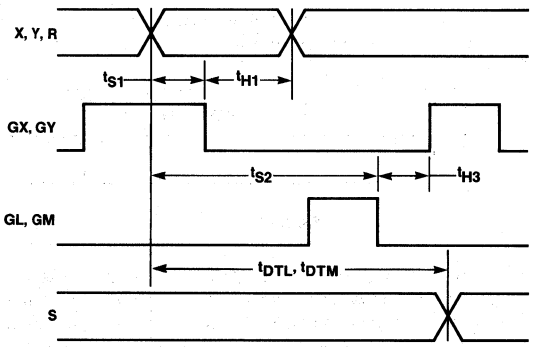


Figure 4c.

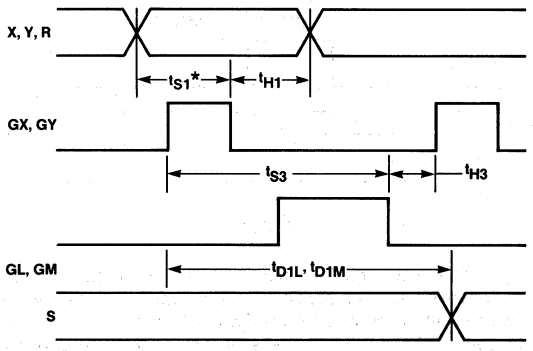


Figure 4d.

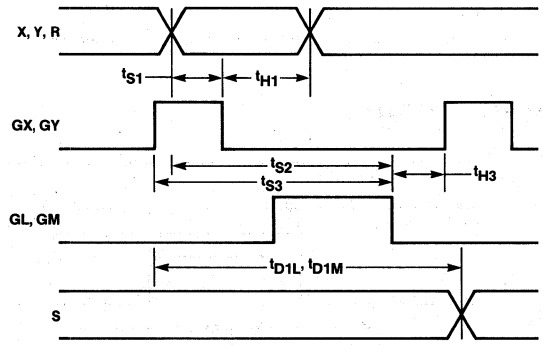


Figure 4e.

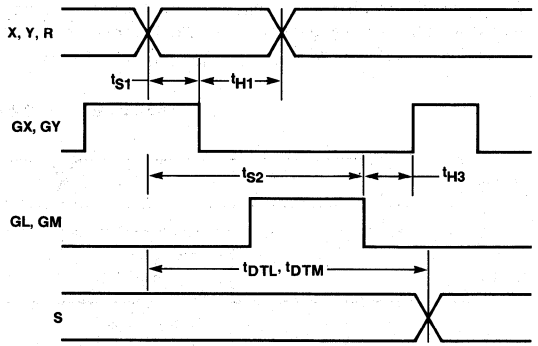


Figure 4f.

* With this particular timing setup time t_{S1} will be automatically met.

The gated multiply represents the pipelined input and output operation. The latch enable lines GX, GY, GL, GM are used to store incoming operands and outgoing results. The particular set-up times that must be met and the time the result takes to reach the outputs depends on two timing relationships. The first is when the rising edges of GX and GY occur with respect to the operand inputs changing, and the second is when the rising edges of GL and GM occur with respect to the rising edges of GX and GY. On the above timing diagrams, denote the absolute time that the operand inputs change as T_{XYR}, the absolute time that the rising edges of GX and GY occur as T_{GXY}, and the absolute time that the rising edges of GL and GM occur as T_{GLM}. Thus, the two delays of concern can be explicitly stated as (T_{GXY} - T_{XYR}) and (T_{GLM} - T_{GXY}). Notice that either of these quantities can be positive or negative depending on which event occurs first. Timing for gated multiplies can then be summarized in the following table:

$T_{GXY} - T_{XYR}$	$T_{GLM} - T_{GXY}$	FIGURE	WHICH SET-UP TIMES MUST BE MET	WHEN RESULT IS PRESENT AT OUTPUTS
$T_{GXY} - T_{XYR} \geq 0$	$T_{GLM}-T_{GXY} \geq t_{S3min}-t_{Wmin}$	4a	t_{S3}	$T_{GLM} + t_{D2}$
$0<T_{XYR}-T_{GXY} \leq t_{Wmin}-t_{S1min}$	$T_{GLM}-T_{GXY} \geq t_{S3min}-t_{Wmin}$	4b	t_{S1}, t_{S2}, t_{S3}	$T_{GLM} + t_{D2}$
$t_{Wmin} - t_{S1min} < T_{XYR} - T_{GXY}$	$T_{GLM}-T_{GXY} \geq t_{S3min}-t_{Wmin}$	4c	t_{S1}, t_{S2}	$T_{XYR} + (t_{DTL}, t_{DTM})^*$
$T_{GXY} - T_{XYR} \geq 0$	$T_{GLM}-T_{GXY} < t_{S3min}-t_{Wmin}$	4d	t_{S3}	$T_{GXY} + (t_{D1L}, t_{D1M})^*$
$0<T_{XYR}-T_{GXY} \leq t_{Wmin}-t_{S1min}$	$T_{GLM}-T_{GXY} < t_{S3min}-t_{Wmin}$	4e	t_{S1}, t_{S2}, t_{S3}	$T_{GXY} + (t_{D1L}, t_{D1M})^*$
$t_{Wmin} - t_{S1min} < T_{XYR} - T_{GXY}$	$T_{GLM}-T_{GXY} < t_{S3min}-t_{Wmin}$	4f	t_{S1}, t_{S2}	$T_{XYR} + (t_{DTL}, t_{DTM})^*$

* For the least and most significant halves of the product respectively.

NOTE: T_{XYR} represents the absolute time when the operand inputs change.
T_{GXY} and T_{GLM} represent the absolute times when the rising edges of the latch controls occur.

Three State Timing

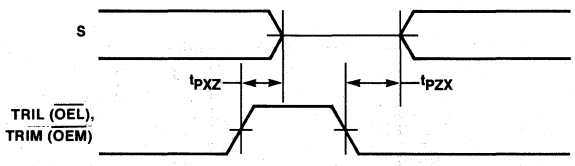


Figure 5.

Test Waveforms

TEST	V_X		OUTPUT WAVEFORM — MEAS. LEVEL
All t_{PD}	5.0V		V_{OH}, V_{OL}, 1.5V
t_{PXZ}	t_{PHZ}	0	V_{OH}, 0.5V, 2.8V
	t_{PLZ}	5	0.5V, V_{OL}, 0.0V
t_{PZX}	t_{PZH}	0	2.8V, 1.5V, V_{OH}
	t_{PZL}	5	0.0V, V_{OL}

Latch Enable Pulse Width (GL, GM, GX, GY)

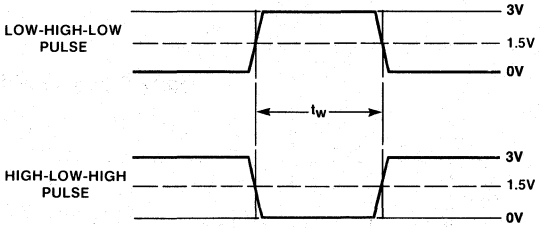


Figure 6.

Load Test Circuit

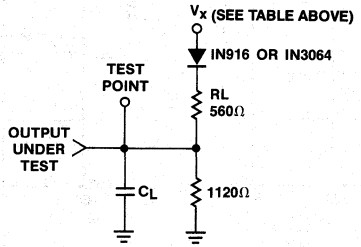


Figure 7.

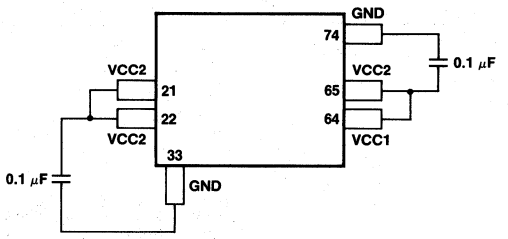
Recommended Bypass Capacitors

The switching currents when the outputs change can be fairly high, and bypass capacitors are recommended to adequately decouple the VCC and GND connections.

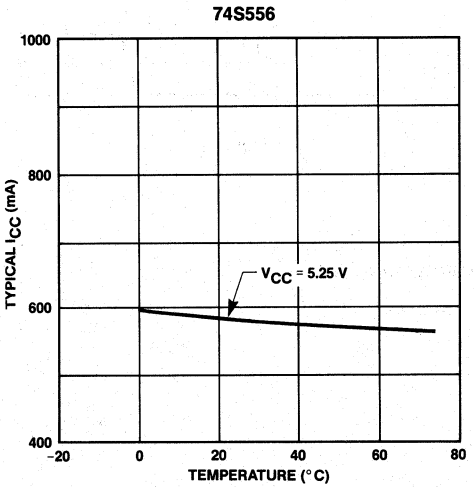
For example, on the 84-terminal LCC package, pins 21 and 22 are VCC2 supplies and should be decoupled with pin 33, a GND input, using a 0.1 μf monolithic ceramic disk capacitor. The capacitor must have good high-frequency characteristics. Also pins 64 and 65, VCC1 and VCC2, should be decoupled with pin 74, a GND input, with a similar capacitor arrangement.

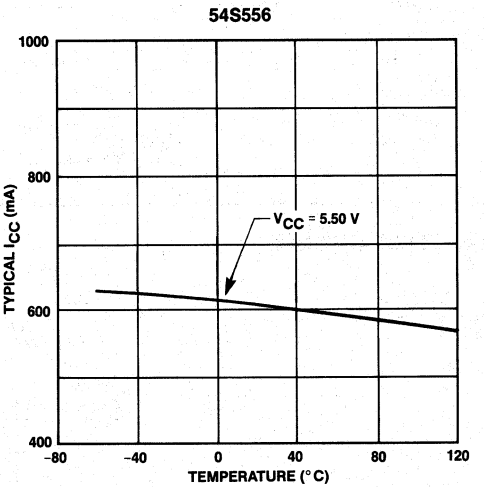
For the 88-pin-grid-array package pins 21 and 22 are VCC2 supplies and should be decoupled with pin 35, the GND pin. Pins 66 and 67, VCC1 and VCC2, should be decoupled with pin 77, the GND pin.

Decoupling Capacitors Shown with the 84-Terminal LCC Package

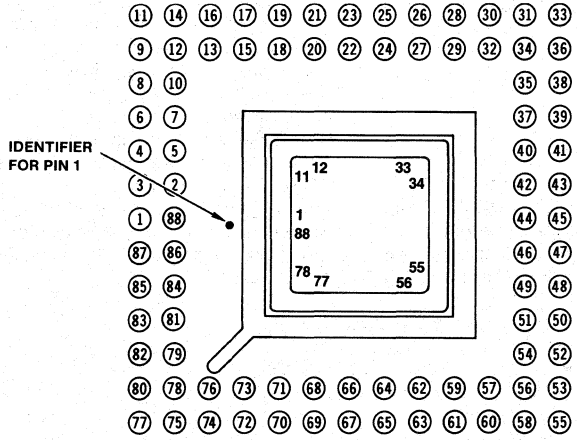


Typical Supply Current Over Temperature Range





88 Pin-Grid-Array
Pin Locations
Bottom View



Pin-Guide For Pin Grid Array

Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name
1	X9	23	N/C*	45	S25	67	VCC2†
2	X10	24	Y8	46	S24	68	N/C*
3	X11	25	Y9	47	S23	69	S7
4	X12	26	Y10	48	S22	70	S6
5	X13	27	Y11	49	S21	71	S5
6	X14	28	Y12	50	S20	72	S4
7	X15	29	Y13	51	S19	73	S3
8	XM	30	Y14	52	S18	74	S2
9	GX	31	Y15	53	S17	75	S1
10	RS	32	YM	54	S16	76	S0
11	RU	33	GY	55	GND	77	GND
12	GND	34	N/C*	56	TRIL ($\overline{\text{OEL}}$)	78	N/C*
13	Y0	35	GND	57	GL	79	GND
14	Y1	36	TRIM ($\overline{\text{OEM}}$)	58	S15	80	X0
15	Y2	37	GM	59	S14	81	X1
16	Y3	38	$\overline{\text{S31}}$	60	S13	82	X2
17	Y4	39	S31	61	S12	83	X3
18	Y5	40	S30	62	S11	84	X4
19	Y6	41	S29	63	S10	85	X5
20	Y7	42	S28	64	S9	86	X6
21	VCC2†	43	S27	65	S8	87	X7
22	VCC2†	44	S26	66	VCC1††	88	X8

* Not connected. † VCC2 = Logic VCC. †† VCC1 = Output buffer VCC.

Rounding

Multiplication of two n-bit operands results in a 2n-bit product†. Therefore, in a pure n-bit system it is necessary to convert the double-length product into a single-length product. This can be accomplished by truncating or rounding. The following examples illustrate the difference between the two conversion techniques in decimal arithmetic:

39.2 → 39
39.6 → 39 Truncating

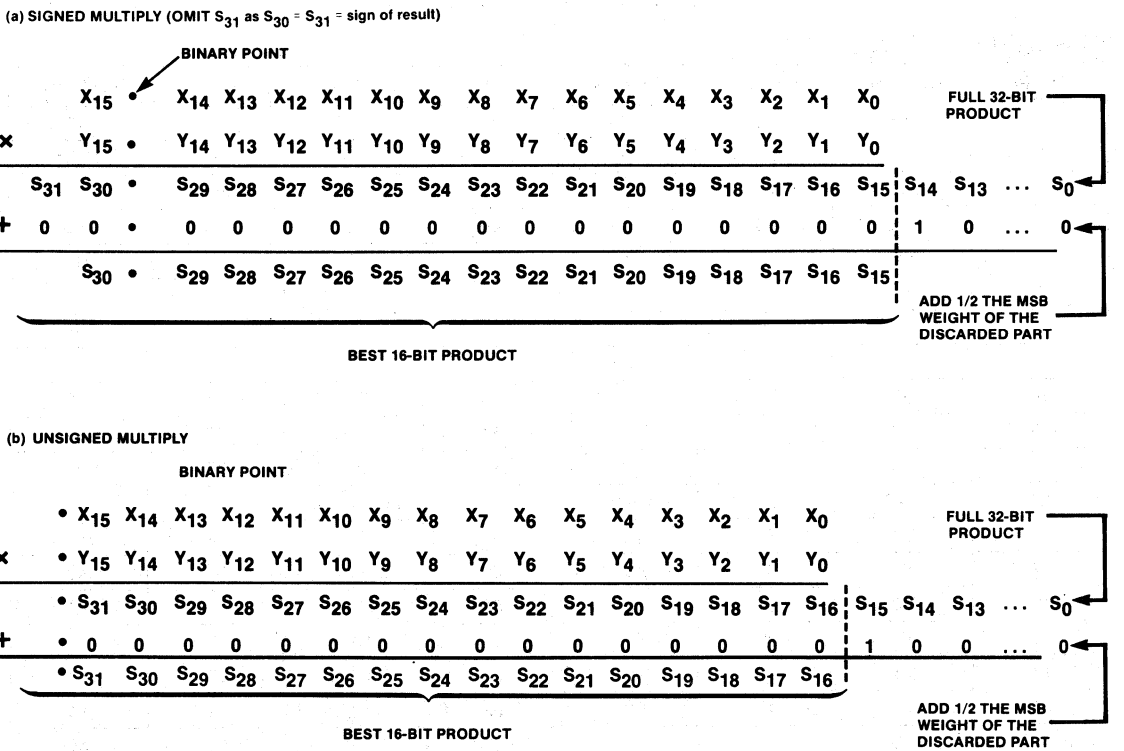
39.2 + 0.5 = 39.7 → 39
39.6 + 0.5 = 40.1 → 40 Rounding

Obviously, rounding maintains more precision than truncating, but it may take one more step to implement. The additional step involves adding one-half of the weight of the single-length LSB to the MSB of the discarded part; e.g., in decimal arithmetic rounding 39.28 to one decimal point is accomplished by adding 0.05 to the number and truncating the LSB:

39.28 + 0.05 = 39.33 → 39.3

The situation in binary arithmetic is quite similar, but two cases need to be considered; signed and unsigned data representation. In signed multiplication, the two MSBs of the result are identical, except when both operands are -1; therefore, the best single-length product is shifted one position to the right with respect to the unsigned multiplications. Figure 8 illustrates these two cases for the 16x16 multiplier. In the signed case, adding one-half of the S_{15} weight is accomplished by adding 1 in bit position 14, and in the unsigned case by adding 1 in bit position 15. Therefore, the 'S556 multiplier has two rounding inputs. RS and RU. Thus, to get a rounded single-length result, the appropriate R input is tied to V_{CC} (logic High) and the other R input is grounded. If a double-length result is desired, both R inputs are grounded.

†In general multiplication of an M-bit operand by an N-bit operand results in an (M + N)-bit product.



NOTES:

(a) In signed (twos-complement) notation, the MSB of each operand is the sign bit, and the binary point is to the right of the MSB. The resulting product has a redundant sign bit and the binary point is to the right of the second MSB of the product. The best 16-bit product is from S_{30} through S_{15}, and rounding is performed by adding "1" to bit position S_{14}.

(b) In unsigned notation the best 16-bit product is the most significant half of the product and is corrected by adding "1" to bit position S_{15}.

Figure 8. Rounding the Result of Binary Fractional Multiplication.

Using the 'S556 in a Pipelined Positive-Edge Triggered Clock System

The 'S556 has internal latches which can be used affectively in systems where things happen on positive-going clock edges. This application is an extension of the gated multiply mode shown in Figure 1, in which a 32-bit product can be latched every t_{S3} nsec in the 'S556.

If the signals GX, GY, GM and GL can be derived from the system clock then the latches can almost have the same effect as having a register. The basic philosophy behind the recommended timing is that the input latches are closed when the output latches are open; the outputs are then closed (and have latched results) and new data is presented to the input latches, which are opened. This is shown by the relation between GX, GY and GL, GM in Figure 9. The set-up time t_{S3} is shown as one value but strictly speaking, it is split as t_{S3L} and t_{S3M} for the least significant and most significant half of the product respectively. The value of t_{S3L} is less than t_{S3M}, for applications requiring the least significant bits of the result as fast as possible.

One note of caution is that a design must always meet the set-up and hold times for X_i, R_i with respect to GX and for Y_i with respect to GY.

The result S_i is available t_{D2} after the rising edge of GM and GL.

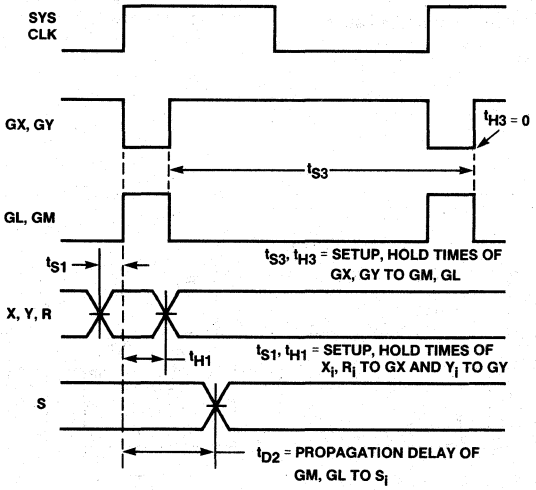


Figure 9.

10

Totally Parallel 32x32 Multiplier

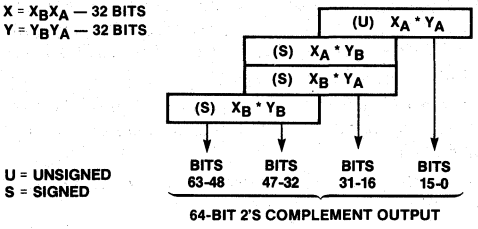
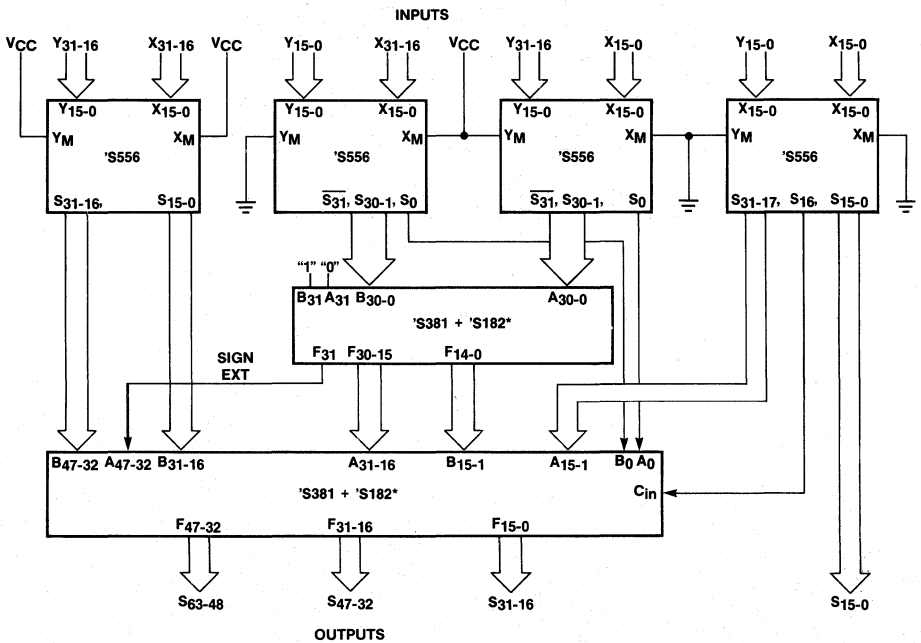


Figure 10. Partial Products for a 32x32 Multiplication

A twos-complement 32x32 multiplication can be performed within 220 nsec using 4 'S556s, 20 'S381s, and 7 'S182s. This 32x32 multiply operation involves adding up four partial products as shown in Figure 10. These four partial products are generated in four multipliers; the outputs are XA*YA, XA*YB, XB*YA, XB*YB, where X31-16 = XB, X15-0 = XA, Y31-16 = YB, Y15-0 = YA.

The implementation of this twos-complement 32x32 multiplier is shown in Figure 11. The outputs of the 16x16 multipliers are connected to two levels of adders to give a 64-bit product. The first level of adders is needed to add the two central partial products of Figure 10, XA*YB and XB*YA. Notice the technique which is used to generate the "sign extension", or the most-significant sum bit of the first level of adders. The 'S556 provides, as a direct output, the complement of the most-significant product bit; having this signal immediately speeds up the sign-extension computation, and reduces the external parts count.



* THESE ARE ADDER BLOCKS USING THE 'S381, A 4-BIT ALU FUNCTION GENERATOR, TO PERFORM A HIGH SPEED ADD OPERATION. THE 'S182 IS A LOOK-AHEAD CARRY GENERATOR WHICH REDUCES THE PROPAGATION DELAY. ALL THE ABOVE PARTS ARE AVAILABLE FROM MONOLITHIC MEMORIES INCORPORATED.

TOTAL MULTIPLY TIME = MULTIPLIER DELAY + ADDER LEVEL 1 DELAY + ADDER LEVEL 2 DELAY = 90 + 65 + 65 = 220 nsec

Figure 11. Implementation of the 32x32 Multiplier

For example, the inputs to the adder in the most significant position are the $\overline{S31}$ outputs from the two central multipliers. The sign extension of the addition of XA*YB and XB*YA is defined as

SIGN EXT = $\overline{\overline{A}.\overline{B}. + \overline{A}.C. + \overline{B}.C.}$, where

A is the most-significant bit of the term XA*YB;
B is the most-significant bit of the term XB*YA; and
C is the carry-in to the most-significant bits of XA*YB and XB*YA, in the adder.

The sign extension can be computed as the negation of the carry-out term of three terms, $\overline{A}$, $\overline{B}$, and C. This term corresponds to the negative of the carry-out of the bit position just one place to the right of the most-significant bit position of the first level of adders. The negative of the carry-out can be generated by presenting a carry-out and a binary "one" to the most significant bit of the adder. The generated sum bit then corresponds to the negation of the carry-out of the previous stage, which is the sign extension required to be added to the 16 most-significant bits of the XB*YB partial product term.

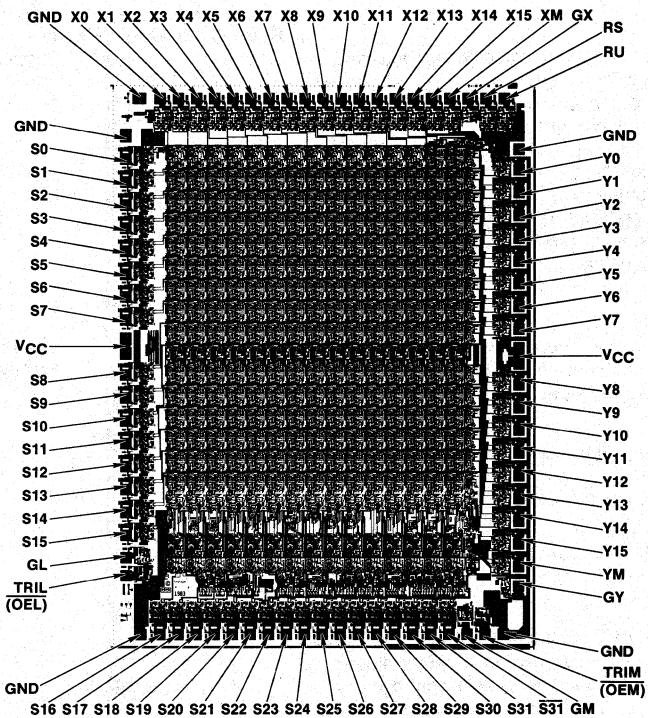
The second level of adders, which performs a 48-bit add function, is fairly straightforward. These adders can be implemented using 'S381 four-bit ALUs and 'S182 carry-bypasses ("carry-lookahead generators") which are available from Monolithic Memories Inc. and from other vendors.

Other configurations such as 48x48 and 64x64 multipliers can be designed using the same methodology, r1.

References

1. "Fast 64x64 Multiplication using 16x16 Flow-through Multiplier and Wallace Trees," Marvin Fox, Chuck Hastings, and Suneel Rajpal, *Monolithic Memories System Design Handbook*, pages 4-77 to 4-84.

Die Configuration



Die Size = 183x243 mil^2

8x8 High Speed Schottky Multipliers
SN54/74S557 SN54/74S558

Features/Benefits

- **Industry-standard 8x8 multiplier**
- **Multiplies two 8-bit numbers; gives 16-bit result**
- **Cascadable; 56x56 fully-parallel multiplication uses only 34 multipliers for the most-significant half of the product**
- **Full 8x8 multiply in 60ns worst case**
- **Three-state outputs for bus operation**
- **Transparent 16-bit latch in 'S557**
- **Plug-in compatible with original Monolithic Memories' 67558**

Description

The 'S557/'S558 is a high-speed 8x8 combinatorial multiplier which can multiply two eight-bit unsigned or signed twos-complement numbers and generate the sixteen-bit unsigned or signed product. Each input operand X and Y has an associated Mode control line, X_M and Y_M respectively. When a Mode control line is at a Low logic level, the operand is treated as an unsigned eight-bit number; whereas, if the Mode control is at a High logic level, the operand is treated as an eight-bit signed twos-complement number. Additional inputs, R_S and R_U, (R, in the 'S557) allow the addition of a bit into the multiplier array at the appropriate bit positions for rounding signed or unsigned fractional numbers.

The 'S557 internally develops proper rounding for either signed or unsigned numbers by combining the rounding input R with X_M, Y_M, $\overline{X_M}$, and $\overline{Y_M}$ as follows:

$R_U = \overline{X_M} \cdot \overline{Y_M} \cdot R$ = Unsigned rounding input to 2^7 adder.

$R_S = (X_M + Y_M)\ R$ = Signed rounding input to 2^6 adder.

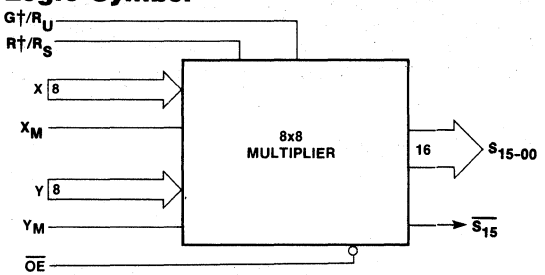
Since the 'S558 has no latches, it does not require the use of pin 11 for the latch enable input G, so R_S and R_U are brought out separately.

The most-significant product bit is available in both true and complemented form to assist in expansion to larger signed multipliers. The product outputs are three-state, controlled by an assertive-low Output Enable which allows several multipliers to be connected to a parallel bus or be used in a pipelined system. The device uses a single +5V power supply and is packaged in a standard 40-pin DIP.

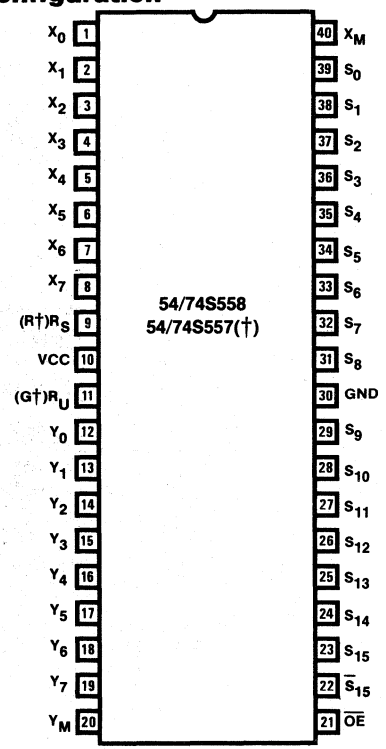
Ordering Information

PART NUMBER	PACKAGE	TEMPERATURE
54S557, 54S558	J, (44), (L)	Military
74S557, 74S558	N,J,	Commercial

Logic Symbol



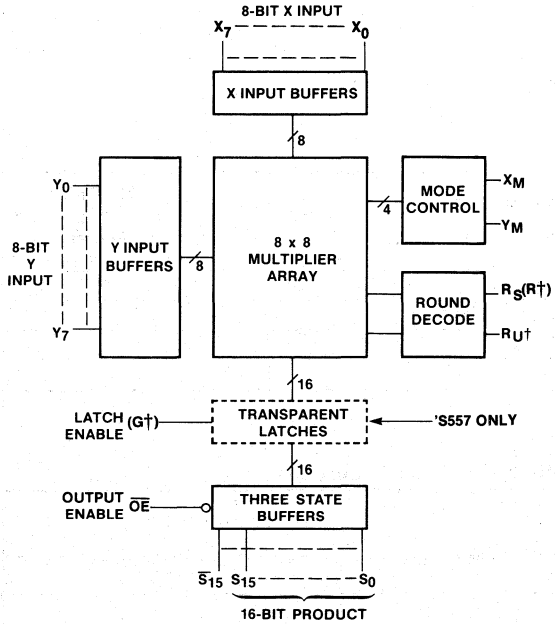
Pin Configuration



†For 54/74S557 Pin 9 is R and Pin 11 is G.

TWX: 910-338-2376
2175 Mission College Blvd. Santa Clara, CA 95054-1592 Tel: (408) 970-9700 TWX: 910-338-2374
Monolithic Memories

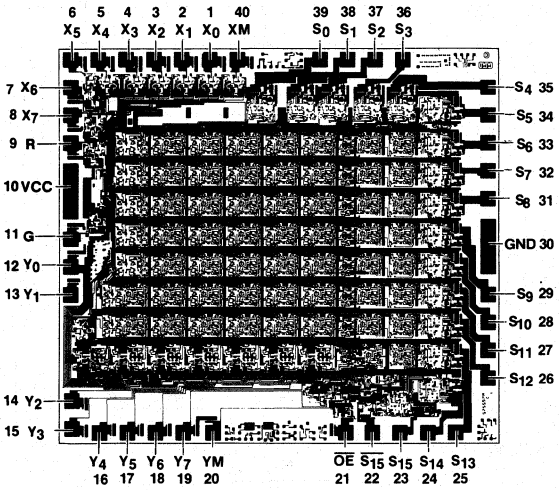
Logic Diagram



†For 54/74S557 Pin 9 is R and Pin 11 is G.

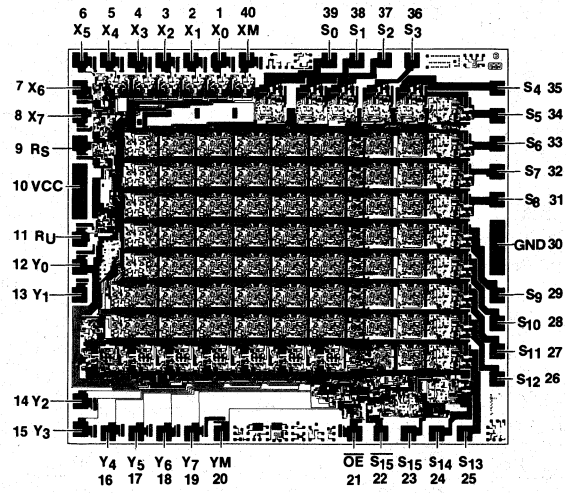
Die Configurations

'S557



Die Size: 144x130 mil^2

'S558



Die Size: 144x130 mil^2

Absolute Maximum Ratings

Supply voltage V_{CC} ..... 7.0 V
Input voltage ..... 7.0 V
Off-state output voltage ..... 5.5 V
Storage temperature ..... −65° to +150°C

Operating Conditions

SYMBOL	PARAMETER	DEVICE	MILITARY MIN	MILITARY TYP	MILITARY MAX	COMMERCIAL MIN	COMMERCIAL TYP	COMMERCIAL MAX	UNITS
V_{CC}	Supply voltage	all	4.5	5	5.5	4.75	5	5.25	V
T_A	Operating free-air temperature	all	−55		125*	0		75	°C
t_{su}	X_i, Y_i to G set	'S557	50			40			ns
t_h	X_i, Y_i to G hold time	'S557	0			0			ns
t_w	Latch enable pulse width	'S557	20			15			ns

* Case temperature

Electrical Characteristics Over Operating Conditions

SYMBOL	PARAMETER	TEST CONDITIONS		MIN	TYP†	MAX	UNIT
V_{IL}	Low-level input voltage					0.8	V
V_{IH}	High-level input voltage			2			V
V_{IC}	Input clamp voltage	V_{CC} = MIN	I_I = −18mA			−1.5	V
I_{IL}	Low-level input current	V_{CC} = MAX	V_I = 0.5V			−1	mA
I_{IH}	High-level input current	V_{CC} = MAX	V_I = 2.4V			100	μA
I_I	Maximum input current	V_{CC} = MAX	V_I = 5.5V			1	mA
V_{OL}	Low-level output voltage	V_{CC} = MIN	I_{OL} = 8mA			0.5	V
V_{OH}	High-level output voltage	V_{CC} = MIN	I_{OH} = −2mA	2.4			V
I_{OZL}	Off-state output current	V_{CC} = MAX	V_O = 0.5V			−100	μA
I_{OZH}			V_O = 2.4V			100	μA
I_{OS}	Output short-circuit current*	V_{CC} = MAX	V_O = 0V	−20		−90	mA
I_{CC}	Supply current	V_{CC} = MAX			200	280	mA

* Not more than one output should be shorted at a time and duration of the short-circuit should not exceed one second.

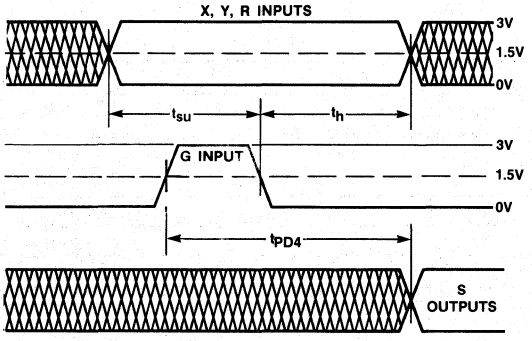
† Typicals at 5.0V V_{CC} and 25°C T_A.

Switching Characteristics Over Operating Conditions

SYMBOL	PARAMETER	DEVICE	TEST CONDITIONS	MILITARY MIN	MILITARY TYP†	MILITARY MAX	COMMERCIAL MIN	COMMERCIAL TYP†	COMMERCIAL MAX	UNIT
t_{PD1}	X_i, Y_i to S_{7-0}	All	C_L = 30pF, R_L = 560Ω, see test figures		40	60		40	50	ns
t_{PD2}	X_i, Y_i to S_{15-8}	All			45	70		45	60	ns
t_{PD3}	X_i, Y_i to $\overline{S}_{15}$	All			50	75		50	65	ns
t_{PD4}	G to S_i	'S557			20	40		20	35	ns
t_{PXZ}	$\overline{OE}$ to S_i	All			20	40		20	30	ns
t_{PZX}	$\overline{OE}$ to S_i	All			15	40		15	30	ns

Timing Waveforms

Setup and Hold Times ('S557)

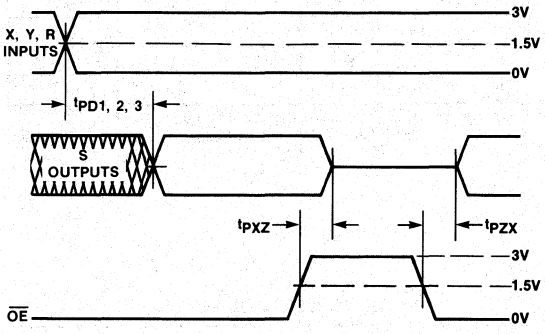


NOTE: If the rising edge of G occurs *before* ($t_{SU_{MIN}}$-$t_{W_{MIN}}$) from the inputs changing, then the applicable propagation delays are t_{PD}, t_{PD2} and t_{PD3}, (and not t_{PD4}). In this case the time at which the results arrive at the outputs depends on when the inputs change instead of when the rising edge of G occurs.

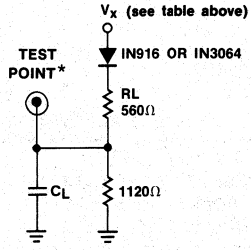
Test Waveforms

TEST	V_X		OUTPUT WAVEFORM — MEAS. LEVEL
All t_{PD}	5.0V		V_{OH}, V_{OL}, 1.5V
t_{PXZ}	for t_{PHZ}	for t_{PLZ}	V_{OH}, 0.5V, 0.5V, V_{OL}, 2.8V, 0.0V
	0.0V	5.0V	
t_{PZX}	for t_{PZH}	for t_{PZL}	2.8V, 0.0V, 1.5V, V_{OH}, V_{OL}
	0.0V	5.0V	

Propagation Delay

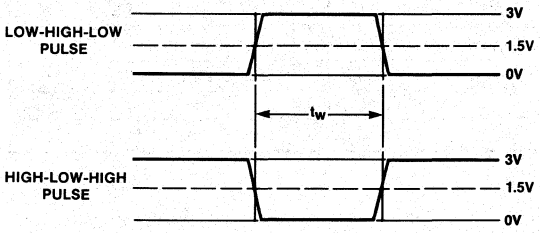


Test Load



* The "TEST POINT" is driven by the output under test, and observed by instrumentation.

Latch-Enable Pulse Width ('S557)



Definition of Timing Diagram

WAVEFORM	INPUTS	OUTPUTS
	DON'T CARE; CHANGE PERMITTED	CHANGING; STATE UNKNOWN
	NOT APPLICABLE	CENTER LINE IS HIGH IMPEDANCE STATE
	MUST BE STEADY	WILL BE STEADY

10

SUMMARY OF SIGNALS/PINS	
X_7-X_0	Multiplicand 8-bit data inputs
Y_7-Y_0	Multiplier 8-bit data inputs
X_M, Y_M	Mode control inputs for each data word; LOW for unsigned data and HIGH for twos-complement data
S_{15}-S_0	Product 16-bit output
$\overline{S}_{15}$	Inverted MSB for expansion
R_S, R_U	Rounding inputs for signed and unsigned data, respectively ('S558 only)
G	Transparent latch enable ('S557 only)
$\overline{OE}$	Three-state enable for S_{15}-S_0 and $\overline{S_{15}}$ outputs
R	Rounding input for signed or unsigned data; combined internally with X_M, Y_M ('S557 only)

74S557 FUNCTION TABLE

INPUTS		PRODUCT RESULT FROM ARRAY	LATCH CONTENTS (INTERNAL TO PART)	OUTPUTS	FUNCTION
$\overline{OE}$	G	T_i	Q_i	S_i	
L	L	X	L	L	Latched
L	L	X	H	H	
L	H	L	(L)*	L	Transparent
L	H	H	(H)*	H	
H	L	X	(L)	Z	Hi-Z; Latched Data not Changed
H	L	X	(H)	Z	
H	H	X	(X)*	Z	Hi-Z

*Identical with product result passing through latch.

ROUNDING INPUTS

'S557

INPUTS			ADDS	
X_M	Y_M	R	2^7	2^6
L	L	H	YES	NO
L	H	H	NO	YES
H	L	H	NO	YES
H	H	H	NO	YES
X	X	L	NO	NO

'S558

INPUTS		ADDS		USUALLY USED WITH	
R_U	R_S	2^7	2^6	X_M	Y_M
L	L	NO	NO	X	X
L	H	NO	YES	H†	H†
H	L	YES	NO	L	L
H	H	YES	YES	*	*

†In mixed mode, one of these could be Low but not both.
*Usually a nonsense operation. See applications section of data sheet.

MODE CONTROL INPUTS

OPERATING MODE	INPUT DATA		MODE CONTROL INPUTS	
	X_7-X_0	Y_7-Y_0	X_M	Y_M
Unsigned	Unsigned	Unsigned	L	L
Mixed	Unsigned	Twos-Comp.	L	H
	Twos-Comp.	Unsigned	H	L
Signed	Twos-Comp.	Twos-Comp.	H	H

Functional Description

The 'S557 and 'S558 multipliers are 8x8 full-adder Cray arrays capable of multiplying numbers in unsigned, signed, twos-complement, or mixed notation. Each 8-bit input operand X and Y has associated with it a mode control which determines whether the array treats this number as signed or unsigned. If the mode control is at High logic level, then the operand is treated as a twos-complement number with the most-significant bit having a negative weight; whereas, if the mode control is at a Low logic level, then the operand is treated as an unsigned number.

The multiplier provides all 16 product bits generated by the multiplication. For expansion during signed or mixed multiplication the most-significant product bit is available in both true and complemented form. This allows an adder to be used as a subtractor in many applications and eliminates the need for certain SSI circuits.

Two additional inputs to the array, R_S and R_U, allow the addition of a bit at the appropriate bit position so as to provide rounding to the best signed or unsigned fractional eight-bit result. These inputs can also be used for rounding in larger multipliers. In the 'S557, these two inputs are generated internally from the mode controls and a single R input.

The product outputs of the multiplier are controlled by an assertive-low Output Enable control. When this control is at a Low logic level the multiplier outputs are active, while if the control is at a High logic level then the outputs are placed in a high-impedance state. This three-state capability allows several multipliers to drive a common bus, and also allows pipelining of multiplication for higher-speed systems.

Rounding

Multiplication of two n-bit operands results in a 2n-bit product†. Therefore, in an n-bit system it is necessary to convert the double-length product into a single-length product. This can be accomplished by truncating or rounding. The following examples illustrate the difference between the two conversion techniques in decimal arithmetic:

$$\left.\begin{array}{l} 39.2 \rightarrow 39 \\ 39.6 \rightarrow 39 \end{array}\right\} \text{ Truncating}$$

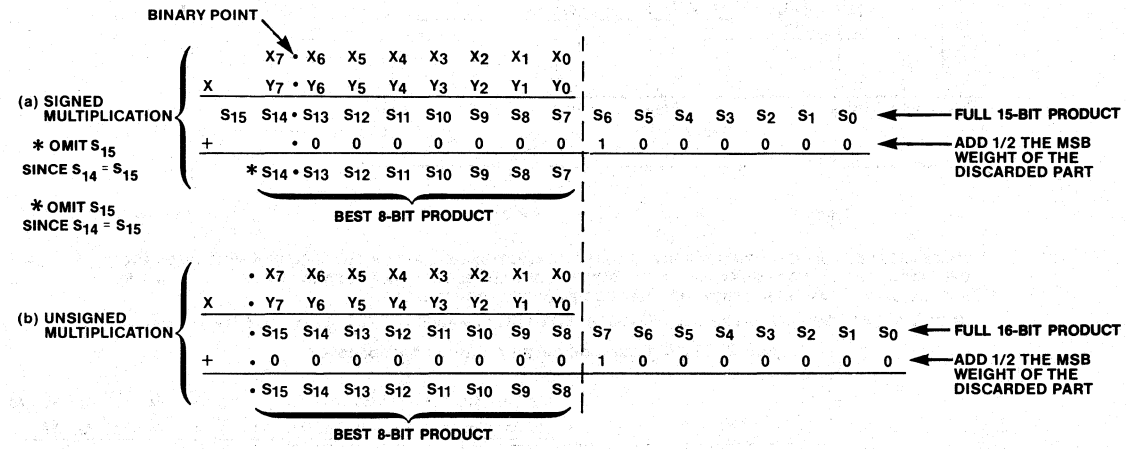
$$\left.\begin{array}{l} 39.2 + 0.5 = 39.7 \rightarrow 39 \\ 39.6 + 0.5 = 40.1 \rightarrow 40 \end{array}\right\} \text{ Rounding}$$

Obviously, rounding maintains more precision than truncating, but it may take one more step to implement. The additional step involves adding one-half of the weight of the single-length LSB to the MSB of the discarded part; e.g., in decimal arithmetic rounding 39.28 to one decimal point is accomplished by adding 0.05 to the number and truncating the LSB:

$$39.28 + 0.05 = 39.33 \rightarrow 39.3$$

The situation in binary arithmetic is quite similar, but two cases need to be considered: signed and unsigned data representation. In signed multiplication, the two MSBs of the result are identical, except when both operands are −1; therefore, the best single-length product is shifted one position to the right with respect to the unsigned multiplications. Figure 1 illustrates these two cases for the 8x8 multiplier. In the signed case, adding one-half of the S_7 weight is accomplished by adding 1 in bit position 6, and in the unsigned case 1 is added to bit position 7. Therefore, the 'S558 multiplier has two rounding inputs, R_S and R_U. Thus, to get a rounded single-length result, the appropriate R input is tied to V_{CC} (logic High) and the other R input is grounded. If a double-length result is desired, both R inputs are grounded for the 'S558, and the single R input is grounded for the 'S557.

†In general: multiplication of an M-bit operand by an N-bit operand results in an (M + N)-bit product.



NOTES:

(a) In signed (twos-complement) notation, the MSB of each operand is the sign bit, and the binary point is to the right of the MSB. The resulting product has a redundant sign bit and the binary point is to the right of the second MSB of the product. The best eight-bit product is from S_{14} through S_7, and rounding is performed by adding "1" to bit position S_6.

(b) In unsigned notation the best 8-bit product is the most significant half of the product and is corrected by adding "1" to bit position S_7.

Figure 1. Rounding the Result of Binary Fractional Multiplication

Signed Expansion

The most-significant product bit has both true and complement outputs available. When building larger signed multipliers, the partial products (except at the lower stages) are signed numbers. These unsigned and signed partial products must be added together to give the correct signed product. Having both the true and complemented form of the most-significant product bit available assists in this addition. For example, say that two signed partial products must be added and MSI adders are used; we then have the situation of adding together the carry from the previous adder stage plus the addition of the two negative most-significant partial-product bits. The result of adding these variables must be a positive sum and a negative carry (borrow). The equations for this are:

$S = A \oplus B \oplus C$

$C_{OUT} = AB + BC + CA$

where C is the carry-in and A and B are the sign bits of the two partial products.

Now an adder produces the equations:

$S = A \oplus B \oplus C$

$C_{OUT} = AB + BC + CA$

Examining these equations, it can be seen that, if the inversions of A and B are used, then the most significant sum bit of the adder is the sign extension bit.

Sign ext = $AB + B\overline{C} + \overline{C}A = \overline{\overline{A}\overline{B} + \overline{B}C + \overline{C}\overline{A}}$,

and the sum remains the same.

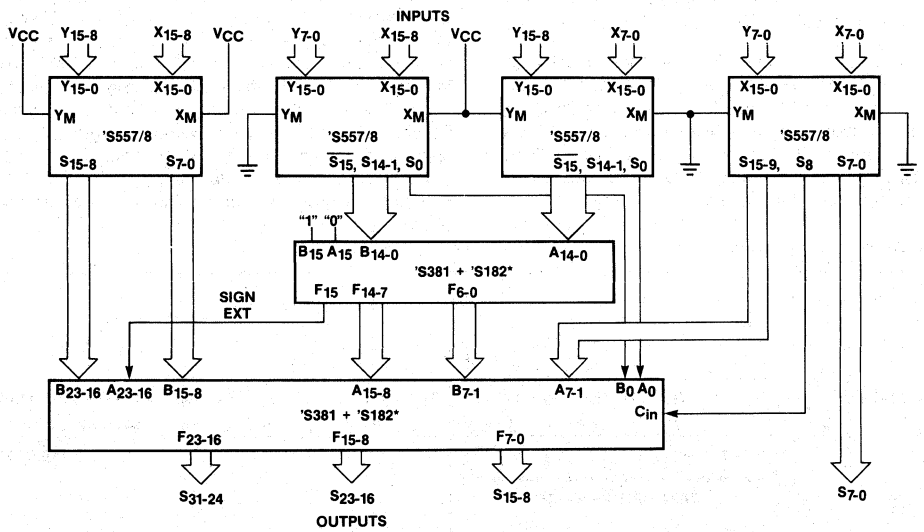
16x16 Twos-Complement Multiplication

The 16-bit X operand is broken into two 8-bit operands (X_7-X_0 and X_{15}-X_8), as is the Y operand. Since the situation is that of a cross-product, four partial products are generated as follows:

$$A = X_L * Y_L$$
$$B = X_L * Y_H$$
$$C = X_H * Y_L$$
$$D = X_H * Y_H$$

where the subscript L stands for bits 7-0, ("low or least-significant half), and the subscript H stands for bits 15-8.

Expanded twos-complement multiplication requires a sign extension of the B and C partial products. Thus, B_{15} and C_{15} need to be extended eight positions to the left (to align with D_{15}). In this approach two more adders are required. But the complement of the MSB ($\overline{S}_{15}$) on the 'S557/8 can be used to save these two adders. Figure 2 shows the implementation of 16x16 signed twos-complement multiplication in this manner.



* THESE ARE ADDER BLOCKS USING THE 'S381, A 4-BIT ALU FUNCTION GENERATOR, TO PERFORM A HIGH-SPEED ADD OPERATION. THE 'S182 IS A LOOKAHEAD CARRY GENERATOR AND REDUCES THE PROPAGATION DELAY. ALL OF THE ABOVE PARTS ARE AVAILABLE FROM MONOLITHIC MEMORIES INCORPORATED.

TOTAL MULTIPLY TIME = MULTIPLIER DELAY + ADDER LEVEL 1 DELAY + ADDER LEVEL 2 DELAY = 60+44+64 = 168 nsec

Figure 2. 16x16 Twos-Complement Signed Multiplication

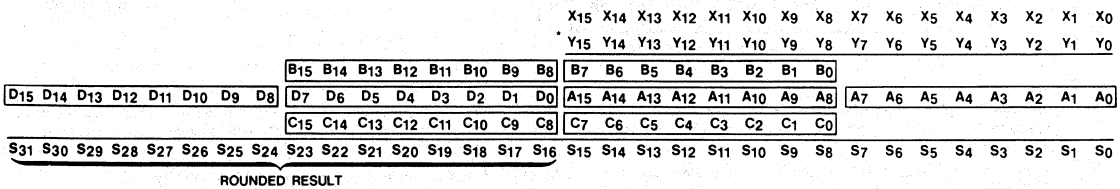


Figure 3. Unsigned Expansions of the 8x8 Multiplier to 16x16 Multiplication

Applications: How to Design Superspeed Cray Multipliers with '558s

by Chuck Hastings

Multiplication, as most of us think of it, is performed by repeated addition and shifting. When we multiply using pencil and paper, according to the familiar elementary-school method, we first write down the multiplicand, and then write down the multiplier immediately under it and underline the multiplier. Then we take the least-significant digit of the multiplier, multiply that digit by the entire multiplicand, and record the answer in the top row of our workspace, underneath the line. Then we repeat, using now the second-least-significant multiplier digit, and record that answer below the first one, pushed one digit position (that is, "shifted") to the left. This process continues until we run out of multiplier digits (or out of patience), at which point we add up the constants of the whole diamond-shaped workspace and record at the bottom an answer which consists of either m + n - 1 digits or m + n digits, where there are m digits in the multiplier and n digits in the multiplicand. An example, voila':

```
  125  (multiplicand)
 x107  (multiplier)
-------
  875  (7 x 125)
  000  (0 x 125, shifted left one digit position)
125    (1 x 125, shifted left two digit positions)
-------
13375  (sum of the above)
```

Figure 4. Decimal Multiplication

The decimal number system has no monopoly on truth — our ancestors simply happened to have ten fingers at the time when someone came up with the idea of counting. Binary numbers, as you know, are more copacetic than are decimal numbers with digital-logic elements, which like to settle comfortably into one voltage state ("High) or another ("Low"), rather than into one of ten different states. So we can repeat the above example using binary numbers, right? First, we convert our multiplicand and multiplier to binary:

$$125_{10} = 01111101_2$$
$$107_{10} = 01101011_2$$

The subscripts 10 and 2 refer to the "base" or "radix" of the number system, 10 for decimal and 2 for binary. (Remember your New Math?) For sneaky reasons to be revealed soon, I've used 8-bit binary numbers, which is one bit more than necessary for my example, and added a leading zero. So, we multiply:

```
  01111101(2) = 125(10)
x 01101011(2) = 107(10)
----------
  01111101
 01111101
00000000
 01111101
00000000
 01111101
 01111101
00000000
----------------
0011010000111111 = 13375(10)
```

Figure 5. Binary Multiplication

I've left off the remarks this time, but they're just like the remarks in the decimal example, at least in principle. Just in case you doubt this answer, I'll convert it back:

```
1      1
1      2
1      4
1      8
1     16
1     32
0      0      (   64)
0      0      (  128)
0      0      (  256)
0      0      (  512)
1   1024
0      0      ( 2048)
1   4096
1   8192
0      0      (16384)
0      0      (32768)
-----
13375
```

Figure 6. Binary-to-Decimal Conversion

Now look carefully at the diamond-shaped array of numbers in the workspace in Figure 5. Each row is either the multiplicand 01111101, or else all zeroes. The 01111101 rows correspond to "1" digits in the multiplier, and the all-zero rows to "0" digits in the multiplier. Life does get simpler in some ways when we switch to binary numbers: "multiplying a multiplier digit by the multiplicand" now means just gating a copy of the multiplicand into that position if the digit is "1," and not doing so if the digit is "0."

Seymour Cray, the master computer designer from Chippewa Falls, Wisconsin, whose career has spanned three companies (Univac, Control Data, and now Cray Research) and many inventions, first observed some time in the late 1950s that *computers* also could actually multiply this way, if one merely provided enough components. This last qualifying remark; in those days when even transistors, let alone integrated circuits, in computers were still a novelty was by no means a trivial one! To prove his point (and satisfy a government contract), Cray designed, and Control Data built, a 48x48 multiplier which operated in one microsecond, about 1960. This multiplier was part of a special-purpose array processor for a classified application, and was so big that a CDC 1604 (then considered a large-scale processor) served as its input/output controller. In principle, such a multiplier at that time would have had to consist of 48 48-bit full adders or "mills," each of which received one input 48-bit number from the outputs of the mill immediately above it in the array, and the other 48-bit number from a gate which either allowed the multiplicand to pass through, or else supplied an all-zero 48-bit number. Actually, these mills have to be somewhat *longer* than 48 bits. Anyway, that is at least 2304 full adders, and in 1960 a full-adder circuit normally occupied one small plug-in circuit card.

A later version of this multiplier, in the CDC 7600 super-computer, could produce one 48x48 product out every 275 nanoseconds on a pipelined basis. The pipelining was asynchronous, and the entire humungous array of adders and gating logic could have up to three different products rippling down it at a given instant!

Back to the 1980s. Monolithic Memories has for several years produced an 8x8 Cray multiplier, the 57/67558, as a single 600-mil 40-pin DIP. After we invented this part, AMD second-sourced it, and by now it has become an industry standard. We now also have faster pin-compatible parts, the 54/74S558 and 54/74S557. Like other West Coast companies 2,000 miles from Wisconsin and Minnesota where Seymour Cray does his inventing, Monolithic Memories previously used the term "combinatorial multiplier" instead of "Cray multiplier" for this type of part. However, "combinatorial multiplier" has nine extra letters and five extra syllables, and also inadvertently implies that the technique involves combinatorial logic rather than arithmetic circuits. Some West Coast designs, including our 67558, use a modified internal array with only half as many full-adder circuits and slightly different interconnections, based on the two-bit "Booth-multiplication" algorithm (see reference 1), plus the "Wallace-tree" or "carry-save adder" technique (see references 2 and 3). Conceptually, however, the entire chip or system continues to operate as a Cray multiplier.

The '558, in particular can be thought of as a static logic network which fits exactly the binary multiplication example of Figure 5. (See now why I insisted on using 8-bit binary numbers?) There are no flipflops or latches whatever in the '558 — it is a "flow-through" device. Its 40 pins are used up as follows:

Use of Pins	Input, Output, or Voltage	Number of Pins
Multiplier	I	8
Multiplicand	I	8
Double-Length Product	O	16
Complement of Most-Significant Bit of Double-Length Product	O	1
3-State Output Enable	I	1
Number-Interpretation-Mode Control	I	2
Rounding Control for Product	I	2
Power and Ground	V	2
		40

Table 1. Use of Pins in the '558

The two number-interpretation-mode control pins, one for the multiplier and one for the multiplicand, allow the format for each of these two 8-bit input numbers to be chosen independently, as follows:

Control Input	Interpretation of 8-bit Input Number
L	8-bit unsigned
H	7-bit plus a sign bit

Table 2. Mode Control Input Encoding

The two rounding control pins allow either integer (right-justified) or fractional (left-justified) interpretation of the 14-bits-plus-sign double-length product of two 7-bits-plus-sign numbers for internal rounding of the double-length result to the most accurate 8-bit number. The control encoding is:

R_S Input	R_U Input	Effect
L	L	Disable Rounding
L	H	Round Unsigned
H	L	Round Signed
H	H	Nonsense (see below)

Table 3. Rounding Control Input Encoding

Rounding is normally disabled if the entire 16-bit double-length product output is to be used. If only an 8-bit subset of this product is to be used, this subset can be either bits 15-8 for unsigned rounding as shown in Figure 7, or bits 14-7 for signed rounding as shown in Figure 8. In either case, a "1" is forced into the '558's internal adder network at the bit position indicated by the arrow; adding a "1" into the bit position *below* the least-significant bit of the final answer has the effect of rounding, as you can see after a little thought. Obviously, forcing a "1" into *both* of these adder positions at the same time is a nonsense operation for most applications — it adds a "3" into the middle of the double-length result.

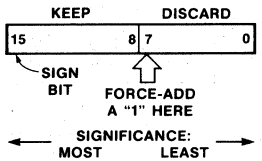


Figure 7. Unsigned Rounding

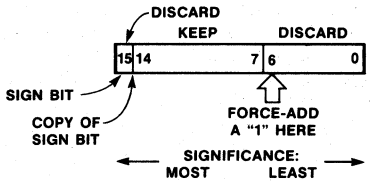


Figure 8. Signed Rounding

By now you probably have a fairly good idea of what a '558 is, and would like a few hints as to how to use it, right? First of all, there is an occasional application in things like video games for *very* fast multiplication, either 8x8 or 16x16, controlled by an 8-bit microprocessor, where there would be one '558 per system (see reference 4). More typically, however, the '558 is a building block, and several of them are used within one system; in fact, maybe more than several — "many." In the usual Silicon-Valley jargon, we can *cascade* a number of '558 (8x8) Cray-multiplier chips to create larger Cray multipliers at the systems level.

For the sake of concreteness, I'll discuss the case of 56x56 multipliers, which are appropriate in floating-point units which deal with "IBM-long-format" numbers which have a 56-bit mantissa. Any computer which emulates, or uses the same floating-point format as, any of the following computers can use such a multiplier:

IBM 360/370
Amdahl 470
Data General Eclipse
Gould/System Engineering SEL 32
Norsk Data 500 (different format)

There are two basic approaches: serial-parallel, and fully parallel. The serial-parallel approach uses seven '558s, and requires seven full multiply-and-add cycles. On the first cycle, the least-significant eight bits of the multiplier are multiplied by the entire multiplicand, and this partial product is saved. On the second cycle, the next-least significant eight bits of the multiplier are multiplied by the multiplicand, and that product (shifted eight bit positions to the left) is added into the first partial product to form the new partial product. And so forth, for five more cycles. It's almost like our decimal-multiplication example of Figure 1, except that instead of base-10 decimal digits we now have base-256 superdigits.

The fully-parallel approach totally applies Cray's usual design philosophy (sometimes characterized as "big, fast, and simple") at the systems level. It uses 49 '558s, in seven ranks; the 'i'th rank performs an operation corresponding to that done during the 'i'th cycle in the serial-parallel implementation. In principle, a complete mill is used to add the outputs of one rank of '558s to those of the rank above it. Or, alternatively, these mills can be laid out in a "tree" arrangement, such as:

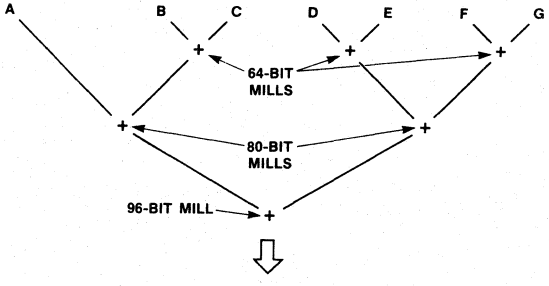


Figure 9. "Tree" Summing Arrangement of Mills for a 56x56 Cray Multiplier

Each letter stands for one rank of '558s, and each "+" stands for a mill of the indicated length. More involved "Wallace-tree" techniques are usually preferable. (See reference 3). If the least-significant half of the double-length product is *never* needed, only 34 'S558s are required. There is one subtlety which needs to be mentioned. If, conceptually, a '558 looks like a diamond —

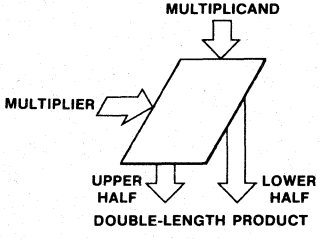


Figure 10. A Single '558 in "Diamond" Notation

then, the 8x56 multiplier for the serial-parallel configuration (which is also one rank of the fully-parallel configuration, which has seven such ranks) looks like this:

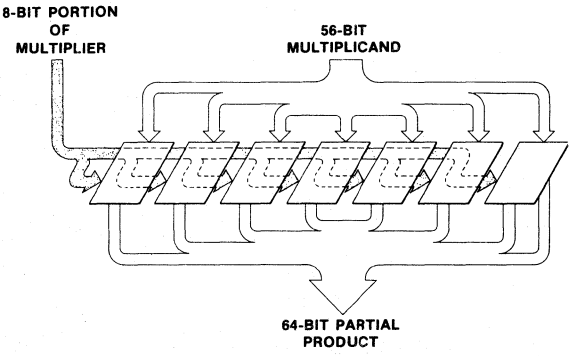


Figure 11. 8x56 Cray Multiplier in "Diamond" Notation

As you may discover after a moment's thought, *each* slanted double line in Figure 8 calls for addition of the outputs of *two* '558s — the eight most significant bits of one, and the eight least-significant bits of the next one to the left. There must also be an extra adder (or at least a "half adder") to propagate the carries from this addition all the way over to the left end of the result. The upshot is that an extra 56-bit mill is needed, in addition to the '558s. The eight least-significant bits of the least-significant '558 do not have to go through this mill, since they do not get added to anything else.

One final note: building up a large Cray-multiplier configuration out of '558s requires a *lot* of full adders, or else a lot of something else equivalent to them. Monolithic Memories also makes the 54/74S381 (a 4-bit "ALU" or "*A*rithmetic *L*ogic *U*nit") and the 54/74S182 (a carry-bypass circuit which works well with the '381); and two faster ALUs, the 54/74F381 and the 54/74F382 are in design. These ALUs and bypasses are excellent building blocks from which to assemble the mills used for summation within a rank of '558s, and also the mills used for tree-summation of the outputs of all ranks. For how to put together one of these mills using '381s, '382s, and '182s, see reference 1. For how to use PROMs as Wallace trees, see reference 3.

Now you can go ahead, design your Cray multiplier out of '558s, and start multiplying full-length numbers together in a fraction of a microsecond. Sound like fun?

References

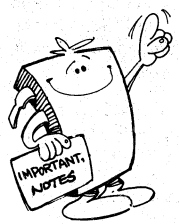
1. "Doing Your Own Thing in High-Speed Digital Arithmetic," Chuck Hastings, Monolithic Memories Conference Proceedings Reprint CP-102

2. "Real-Time Processing Gains Ground with Fast Digital Multiplier," Shlomo Waser and Allen Peterson, *Electronics*, September 29, 1977.

3. "Big, Fast and Simple — Algorithms, Architecture, and Components for High-End Superminis," Ehud "Udi" Gordon and Chuck Hastings, 1982 *Southcon Professional Program*, Orlando, Florida, March 23-25, 1982, paper no. 21/3.

4. "An 8x8 Multiplier and 8-bit μP Perform 16x16-bit Multiplication," Shai Mor, EDN, November 5, 1979, Monolithic Memories Article Reprint AR-109.

NOTE: All of these references are available as application notes from Monolithic Memories Inc.


IMPORTANT
NOTES