16x16 Multiplier/Divider
SN74S516

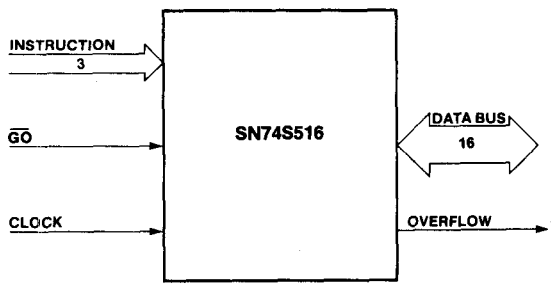
Features/Benefits

- **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**

Ordering Information

PART NUMBER	PACKAGE	TEMPERATURE
SN74S516	24T	Commercial

Logic Symbol



Description

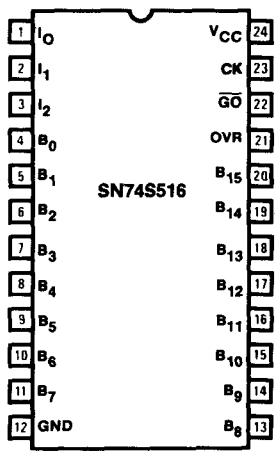
The SN74S516 ('S516) is a bus-organized 16x16 Multiplier/Divider. The device provides both multiplication and division of 2s-complement 16-bit numbers at high speed. There are 28 different multiply options, including: positive and negative multiply, positive and negative accumulation, multiplication 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.

Pin Configuration



SKINNYDIP® is a registered trademark of Monolithic Memories.

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

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

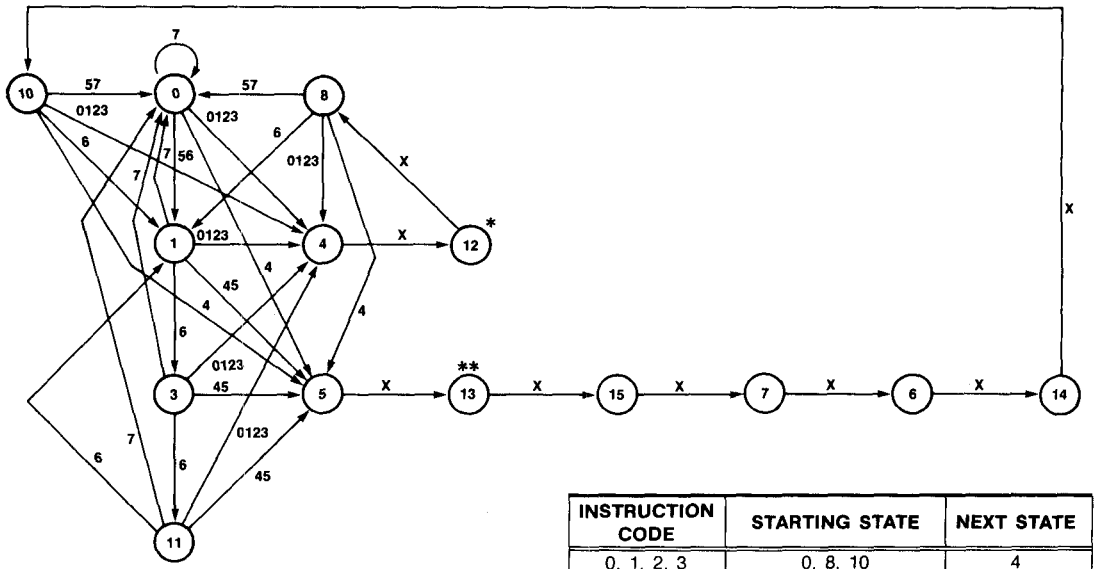
Description (continued)

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.



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

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

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)

OPERATION		TIME-SLOT 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

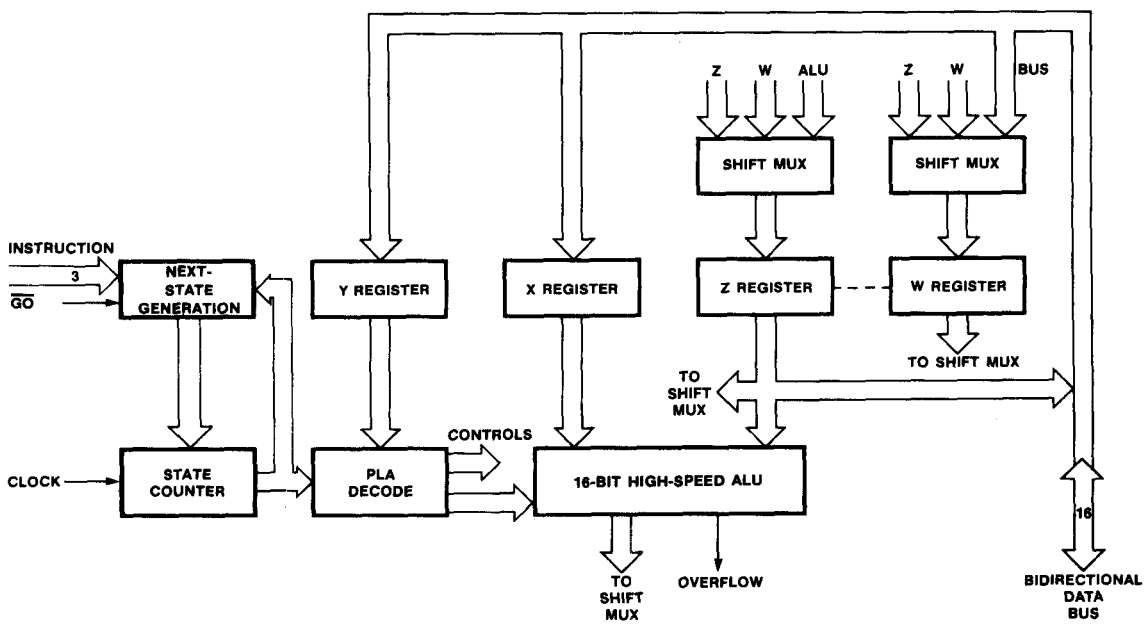


Figure 5. Internal Architecture of the 'S516

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}.

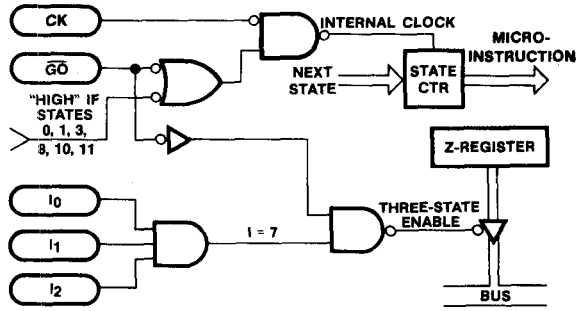


Figure 6. 'S516 Internal Circuitry of "$\overline{GO}$" Line and Three-State-Enable

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.

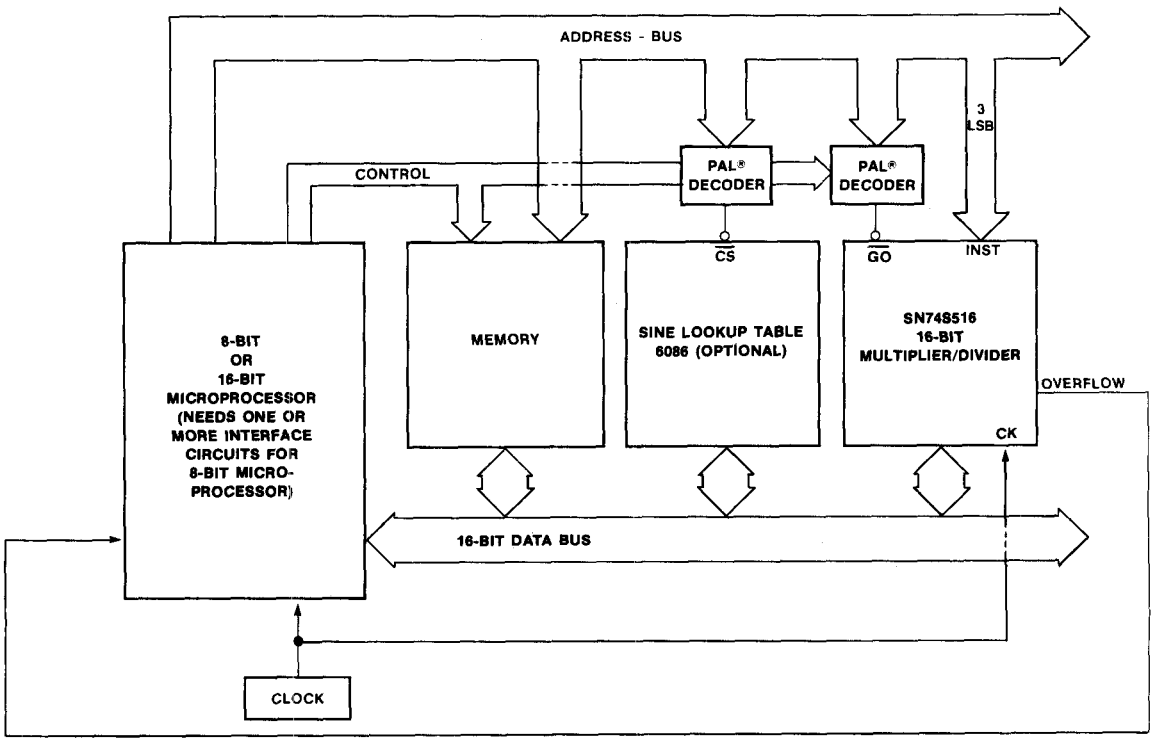


Figure 7. Interfacing the 'S516 to a Microprocessor

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, Multiplier

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		45**	°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.*
** This device has a limited operating temperature range.

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.

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 Circuit" figure below.

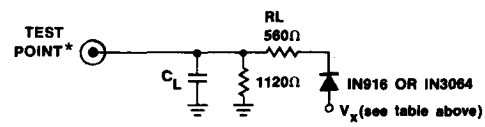
AC Test Conditions

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.

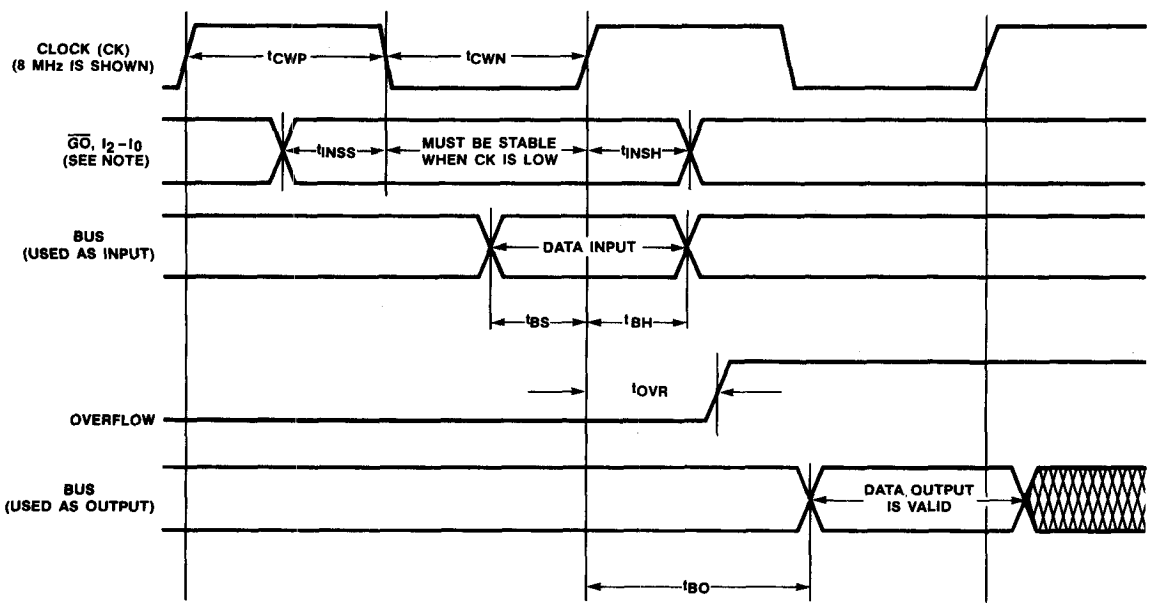
Timing

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

Test Load

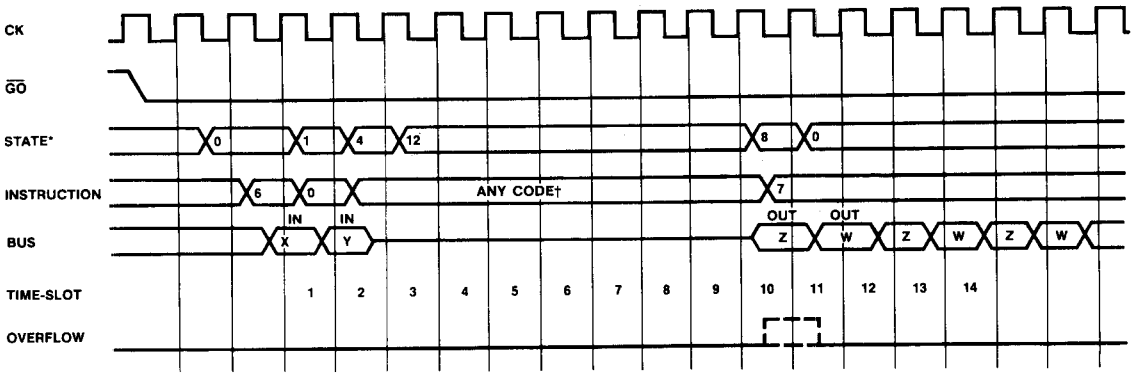


* The "TEST POINT" is driven by the output under test, and observed by instrumentation.



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

Figure 8. Timing Diagram of the 'S516



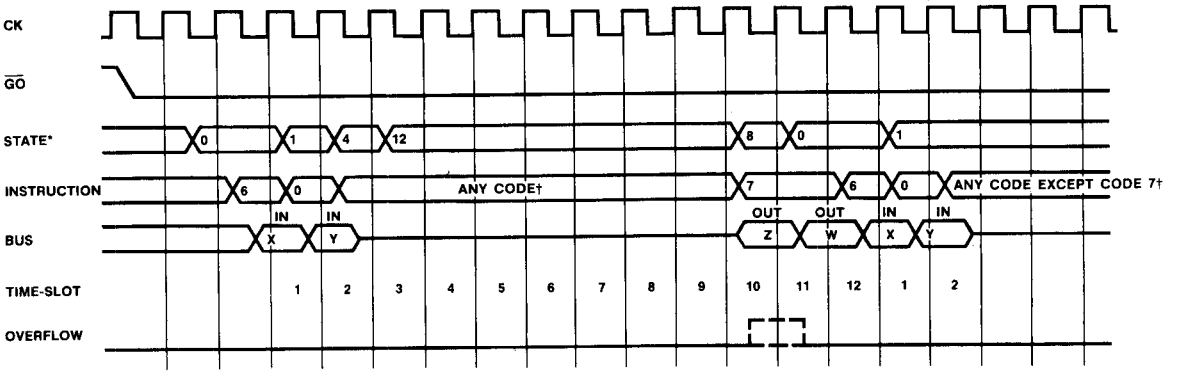
NOTES: Register Z is read at the same time that the overflow signal (if present) is set. If the instruction remains at code 7 after time-slot 11, the contents of registers Z and W are swapped each cycle.

†"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

11

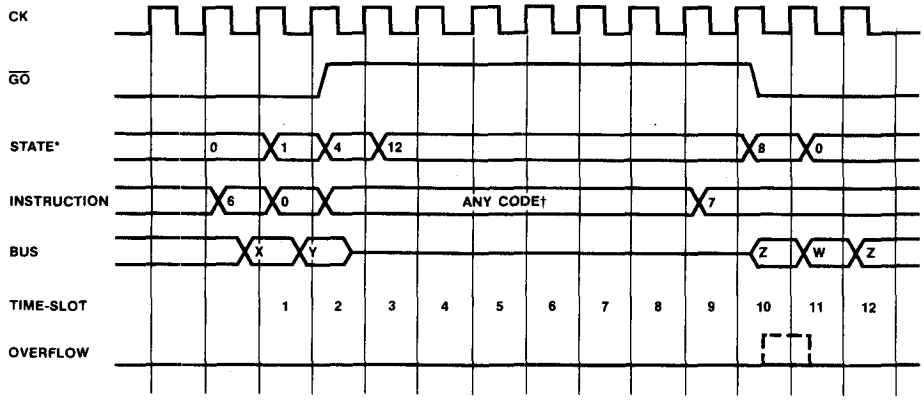


NOTES: The instruction lines may be changed only when CK is high.

†"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.

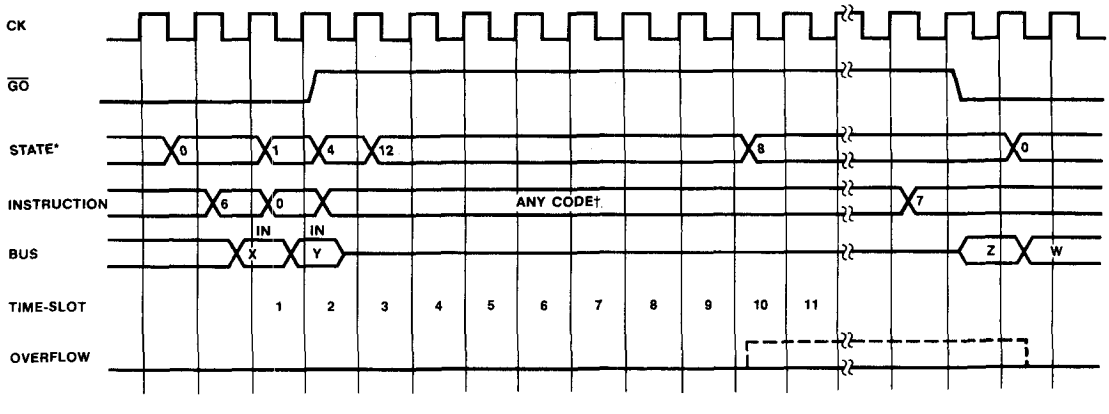
*Not available externally on the 'S516.

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



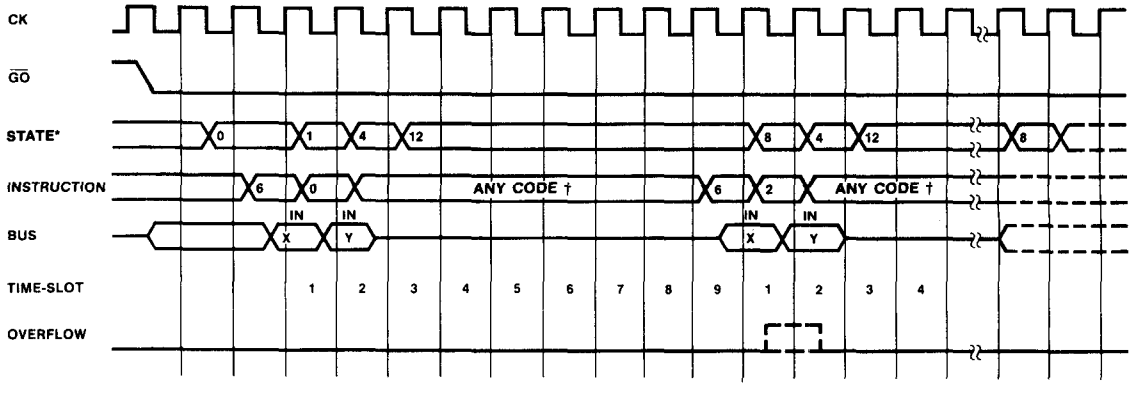
NOTES: Code 7 is given in time-slot 9, but has no effect until time-slot 10 since $\overline{GO}$ is HIGH. After $\overline{GO}$ goes LOW in time-slot 10, Z may be read.
†"Any code" means any of code 0 through code 7.
*Not available externally on the 'S516.

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



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

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 } Perform A·B
	INST X	MULT
	INST X	MULT
	INST X	MULT
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 } Perform C·D + (K_Z, K_W)
	INST X	MULT
	INST X	MULT
	INST X	MULT
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 } Perform E·F + (K_Z, K_W)
	INST X	MULT
	INST X	MULT
	INST X	MULT
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)

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 7	DIVIDE and READ the quotient $Z = \frac{(B, C)}{A}$	
INST 7	READ the remainder W of $\frac{(B, C)}{A}$	

11