SN74S516 Co-Processor Supercharges 68000 Arithmetic

Richard Wm. Blasco, Vincent Coli, Chuck Hastings and Suneel Rajpal

Specialized arithmetic logic, used together with your microprocessor, can provide extra muscle for handling formidable problems like extensive number-crunching operations. In particular, the Monolithic Memories' SN74S516 bipolar multiplier/divider/accumulator can team up with a 16-bit microprocessor such as the 68000 in a co-processor arrangement that significantly improves arithmetic throughput.

The 'S516 uses special hardware and Booth-algorithm techniques to perform multiplication nine times faster than the 68000, and division eight times faster than the 68000. It also includes facilities for performing double-precision arithmetic, and chained operations such as sum-of-products. These capabilities, coupled with the raw speed advantage, permit a number-crunching throughput improvement of 1.7 to 10 times (or more) over 68000-only systems, even when I/O overhead is considered.

The 'S516 is the only bipolar *divider* currently on the market. Its single-bus design and speed are well-matched to 16-bit systems. (However, the 'S516 is also useful in 8-bit systems where 16-bit arithmetic is required.) In general, 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. Even the 68000, a comparatively powerful microprocessor, can benefit.

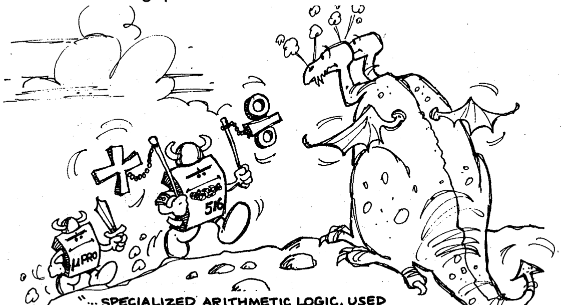
The 68000 and the 'S516 can, therefore, team up to multiply and divide on a bus at an optimum price/performance.

SN74S516 Co-Processor Supercharges 68000 Arithmetic

Richard Wm. Blasco, Vincent Coli, Chuck Hastings and Suneel Rajpal

Why a Co-Processor?

Specialized arithmetic logic, used together with your microprocessor, can provide extra muscle for handling formidable problems like extensive number-crunching operations. In particular, the Monolithic Memories SN74S516 bipolar multiplier/divider/accumulator can team up with a 16-bit microprocessor in a co-processor arrangement that significantly improves arithmetic throughput.

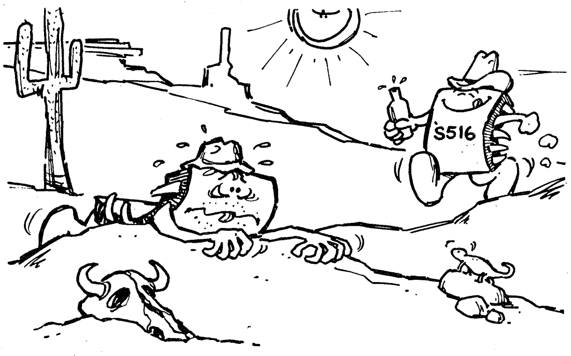


"... SPECIALIZED ARITHMETIC LOGIC, USED TOGETHER WITH YOUR MICROPROCESSOR, CAN PROVIDE EXTRA MUSCLE FOR HANDLING FORMIDABLE PROBLEMS ..."

The 'S516 single-bus design and speed are well matched to 16-bit systems. The 'S516 is also useful in 8-bit systems where 16-bit arithmetic is required. A companion part, the Monolithic Memories SN54/74S508, is optimized for small-scale 8-bit systems.

It will be demonstrated that faster bipolar multipliers would not significantly improve the number-crunching performance provided by the 'S516 in a co-processor environment: they would only cost more.

The 'S516 is the only bipolar divider currently on the market. Newton-Raphson iteration techniques must be used to divide with the high-speed multiply-only chips, slowing them down considerably. The 'S516 is actually faster than these devices when division is performed.



"... THE 74S516 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. Several examples are given in the literature (Refs. 1,2,3). Other existing applications literature for this part describes its use in special-purpose systems (Ref. 4), in a complete co-processor board for a real-time control system (Ref. 5), and in a parallel-architecture signal processor implemented as an array of microprocessors (Ref. 6).

In particular, the S-100 floating-point co-processor board described in Ref. 1 outperforms a dedicated 8086/8087 co-processor arrangement at similar cost, demonstrating the cost-effectiveness of the 'S516 in an 8086-based system.

COST EFFECTIVE!

68000 and 74S516: A perfect match

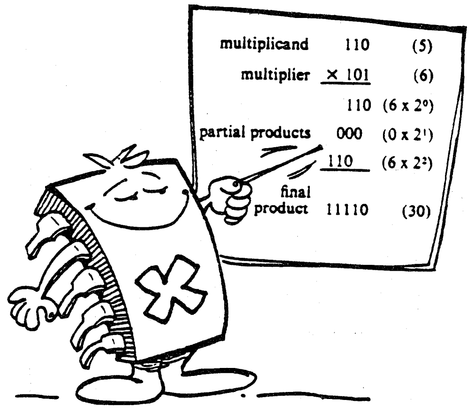
The 68000 16-bit microprocessor and the 74S516 bipolar co-processor have several common characteristics that help the parts work well together. Both parts operate with a 6-MHz system clock and utilize a single bi-directional 16-bit I/O bus. Both parts have internal 32-bit registers. The 'S516 multiplication time is embedded in the 68000 write/read cycle,introducing minimal overhead.

The 6800 and the 'S516 both multiply and divide 16-bit numbers. The similarity ends there. The 68000 multiplies and divides using built-in microcode. Multiplication and division are performed on a bit-by-bit basis. The 68000 multiplication/division speed is achieved by eliminating the many instruction fetches otherwise required.

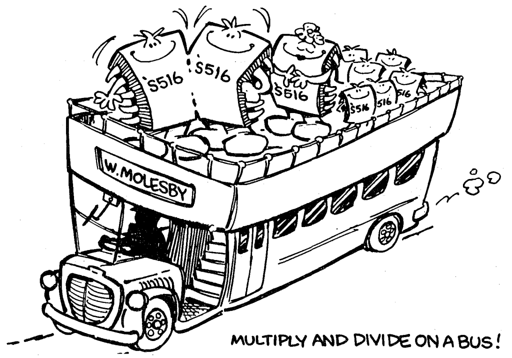
The 'S516 is a dedicated multiplier/divider/accumulator. Special hardware and Booth-algorithm techniques perform multiplication nine times faster than the 68000, and division eight times faster than the 68000. These are raw speed comparisons, excluding I/O, and assume identical clock rates.

Facilities for double-precision arithmetic and chained operations (such as sum-of-products) are built into the 'S516. These capabilities, coupled with the raw speed advantage, permit a number-crunching throughput improvement of 1.7 to 10 times (or more) over 68000-only systems, even when I/O overhead is considered.

Using facilities within the 68000 and the 'S516, floating-point operations could be handled at throughput rates competitive with more expensive MOS floating-point co-processor chips.



The 68000 and the 'S516 can, therefore, team up to multiply and divide on a bus at an optimum price/performance.



74S516 Operation

Detailed operation and interface requirements for both the 68000 and the 74S516 are given in their respective data sheets. This section will point out some of the more relevant items in the 'S516 data sheet.

The logic symbol and the internal architecture of the 'S516 are shown in Figures 1 and 3 respectively. The X register, in Figure 3, is a dual-rank register; it can be preloaded with a new X operand during the previous operation, to enhance the computational speed of the 'S516 where loading of the new X operand and the last multiply/divide cycle using the old X operand occur during the same time slot. Alternatively, if this feature is not used, the X register can be explicitly loaded before each operation.

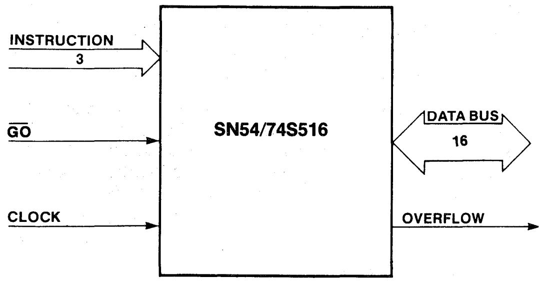


Figure 1. Logic Symbol

INSTRUCTION SEQUENCE				OPERATION	CLOCK CYCLES
				ARITHMETIC OPERATIONS	
			0	$X1 \cdot Y$	9
			1	$-X1 \cdot Y$	9
			2	$X1 \cdot Y + K_Z, K_W$	9
			3	$-X1 \cdot Y + K_Z, K_W$	9
			4	$K_Z, K_W/X1$	21
		5/6	0	$X \cdot Y$	10
		5/6	1	$-X \cdot Y$	10
		5/6	2	$X \cdot Y + K_Z, K_W$	10
		5/6	3	$-X \cdot Y + K_Z, K_W$	10
		5/6	4	K_W/X	22
		5/6	5	K_Z/X	22
	5/6	6	0	$X \cdot Y + Z$	11
	5/6	6	1	$-X \cdot Y + Z$	11
	5/6	6	2	$X \cdot Y + K_Z \cdot 2^{-15}$	11
	5/6	6	3	$-X \cdot Y + K_Z \cdot 2^{-15}$	11
	5/6	6	4	Z, W/X	23
	5/6	6	5	Z/X	23
5/6	6	6	0	$X \cdot Y + Z, W$	12
5/6	6	6	1	$-X \cdot Y + Z, W$	12
5/6	6	6	2	$X \cdot Y + W_{sign}$	12
5/6	6	6	3	$-X \cdot Y + W_{sign}$	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 10 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. Minimum 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+8 for multiplication and n+20 for division.
9. By presenting code 7 on the instruction lines at least one clock cycle before the last clock pulse of the operation cycles, the result (register Z) is available on the bus one clock earlier (see Figure 9 of 'S516 data sheet).
10. 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 2. 'S516 Instruction Set (Partial List)

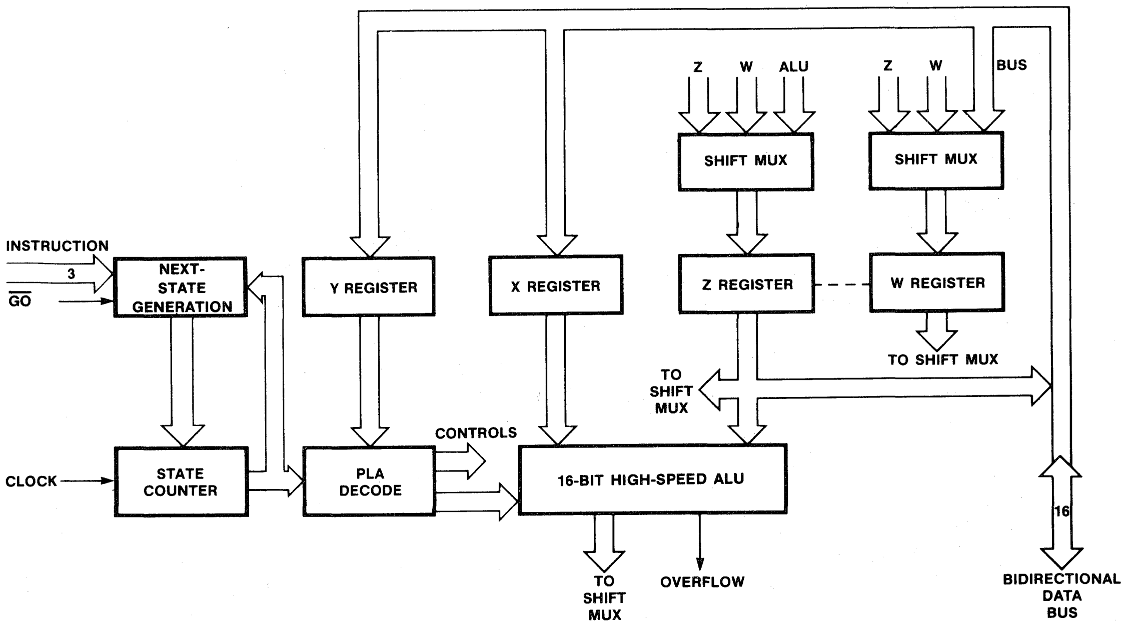


Figure 3. Internal Architecture of the 'S516

There are 28 multiply options and 13 divide options shown in Figure 2. These 41 basic options minimize the external setup and manipulation required when using the 'S516.

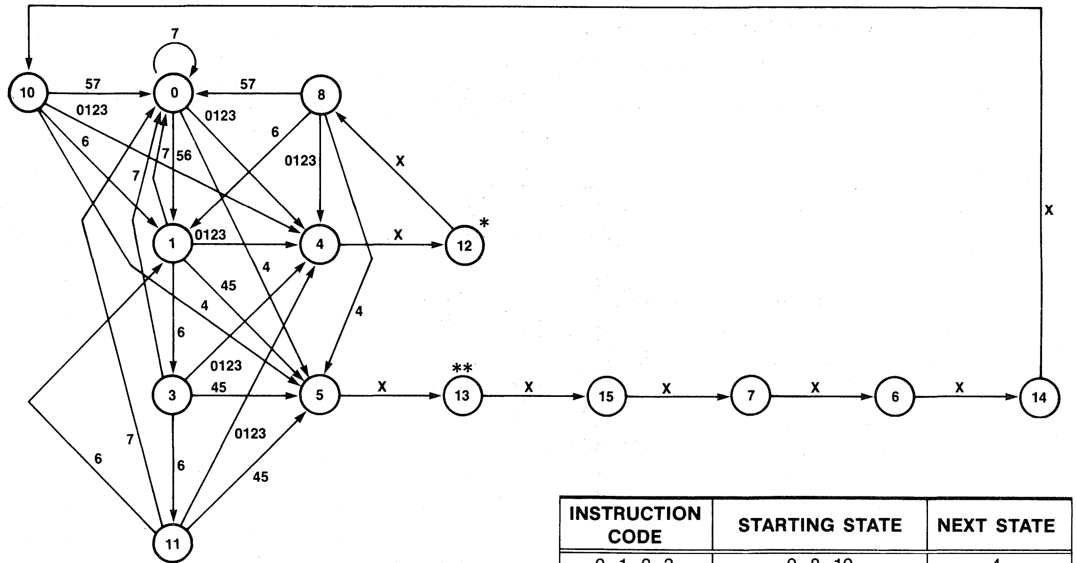
The transition diagram describes the behavior of the 'S516 under a variety of "what if" conditions. State transitions are driven by instructions on the I2-I0 lines, with the internal clock low.

States 0, 8, and 10 represent read states. Repetitive reads (instruction 7) alternate access between the Z(MSW) and W(LSW) output registers. State 0 is the idle state, 8 is the completion of a multiply, and 10 is the completion of a divide. Instruction 5 will round the double-precision Z,W result to 16-bits, with register W set to zero. In state 10, Z holds the quotient and W holds the remainder.

States 1, 3, and 11 represent inputs to the X, Z, and W registers, respectively. The 'S516 can perform integer or fractional signed twos-complement arithmetic. If instruction 5 is used to enter X (transition to state 1), the 'S516 will perform integer multiplication or division. Instruction 6 specifies fractional arithmetic. Fractional arithmetic is useful in floating-point calculations.

States 4 and 5 represent the loading of the Y parameter and the start of the multiplication or division, respectively. The 'S516 requires 8 clock cycles to go from state 4 to state 8, and 20 clock cycles to go from state 5 to state 10. The instruction lines are ignored. The dual-rank X register may be pre-loaded in states 12, 13, 15, 7, 6, or 14.

The cycle-stealing tricks shown in the data sheet may be understood in the light of Figure 4. Cycle-stealing may not be used if instruction 5 is used, as to perform rounding, since this code is handled differently in states 0, 8 and 10.



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 in the 'S516 data sheet for contrasting examples of how this effect operates.

Figure 4. State Transition Diagram

Figure 5 explains the various clock and output gating options. The 'S516 does not have a read/write input. Instruction 7 is a read; all others imply a write operation. The currently-available register (defined by the current state) may be read at any time.

A hardware reset line is not provided. The 'S516 is initialized by holding instruction 7 for at least 21 clock cycles to ensure that the 'S516 has returned to state 0.

The 'S516 data lines may connect directly to the 68000 bus, if loading by other devices is not excessive. The 'S516 data outputs can sink 8 mA. The data inputs use PNP devices equivalent to 1/2 of an LS-TTL load. The other inputs are equivalent to 1/2 of a standard TTL load.

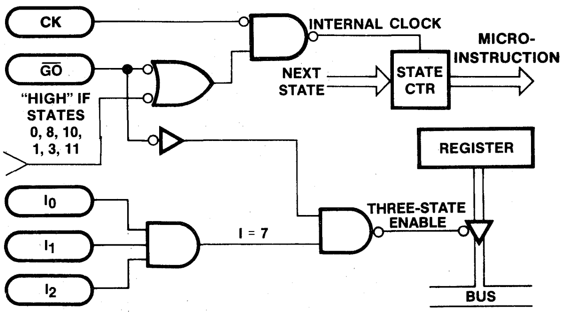


Figure 5. Internal Clock and Output Circuits

A Practical Hardware Interface Circuit

The 68000 is an advanced, high-speed microprocessor with extensive hardware features to support multi-user software. A practical interface circuit should support significant 68000 features, such as error detection and traceback. The multi-user environment means that system crashes (or situations where the 68000 'hangs up') must be avoided. Reliability is essential.

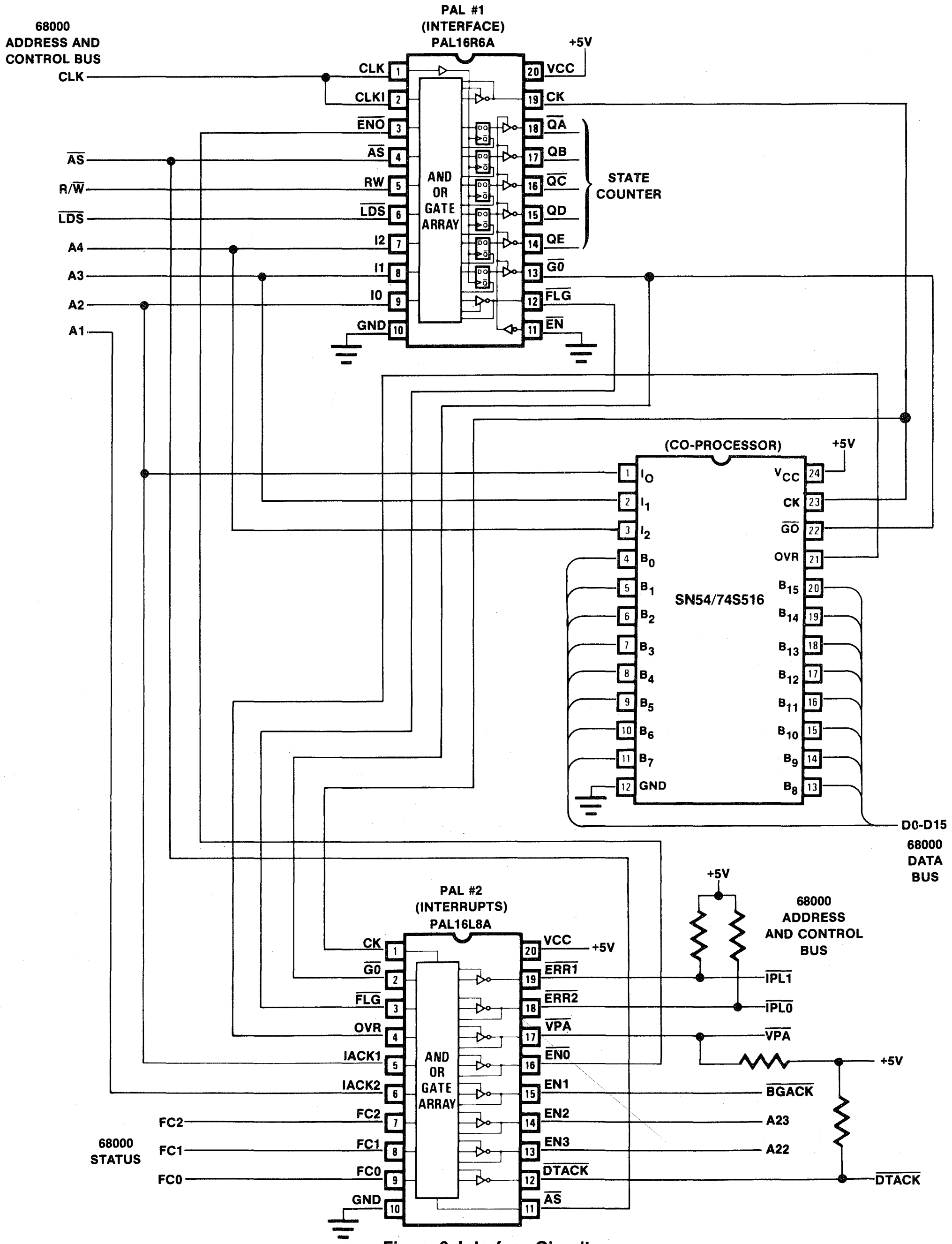


Figure 6. Interface Circuit

The circuit of Figure 6 meets these requirements. It is equivalent to 10-15 TTL SSI/MSI packages. Package count is reduced by the use of high-speed PALs to implement the interface logic.

Monolithic Memories programmable array logic (PAL) chips provide an excellent alternative to TTL when implementing random and control logic. The PAL is similar in concept to the PLA, in that logic is implemented by using an AND-OR matrix array. In the PAL, however, only the AND matrix is programmable. The OR matrix is fixed. This slight reduction in flexibility allows the addition of other functions (registers, bi-directional I/O, etc.) to the basic matrix array, and markedly speeds up the logic propagation time for the circuit.

The PALs are, like the 'S516, bus-oriented. Inputs are equivalent to 1/2 of an LS-TTL load, and outputs can sink 24 mA. Worst-case propagation delay is 25 nsec for the high-speed PALs. The PALs used are available in 20-pin 300-mil packages to minimize board space.

The circuit provides full-featured, reliable operation in a 12-megabyte interrupt-driven 68000 system operating at 6 MHz. DMA devices (such as disk drives) may reside on the bus. Interrupt outputs are provided to indicate error conditions. The interface supports the 68000 autovectoring capability.

PAL #1 keeps track of the internal 'S516 state, provides glitch-free gating of the 'S516 CK input, screens out illegal instruction sequences, and creates a legal transfer acknowledge signal $(\overline{GO})$ and an illegal transfer attempt flag $(\overline{FLG})$.

A more complete 'S516 state emulator is given in APPENDIX A. The complete state emulator can be used to further study the 'S516 features.

PAL #2 decodes the flags provided by PAL #1 and the 'S516 to generate two open-collector interrupts. $\overline{ERR1}$ indicates an 'S516 overflow, while $\overline{ERR2}$ indicates an illegal transfer attempt. The 68000 processor status (FC2-FC0) is decoded to detect an interrupt acknowledge cycle. Independent interrupt acknowledge signals are used to assert the $\overline{VPA}$ ouput if the 'S516 interface generated the interrupt. Assertion of $\overline{VPA}$ initiates the autovector feature of the 68000. ERR2 must be released by accessing the multiplier during the interrupt routine.

Transfer acknowledge $(\overline{DTACK})$ will be asserted for both legal transfers $(\overline{GO})$ or illegal transfers $(\overline{FLG})$. This prevents hangup of the 68000 caused by programming errors. (See Figure 8.)

$\overline{ENO}$ decodes a valid memory-mapped address, using $\overline{BGACK}$ to exclude bus cycles controlled by temporary masters (DMA devices), suppresses interrupt acknowledge cycles, and enables PAL #1.

Address lines A2, A3, and A4 are used to control I0, I1, and I2 respectively. This permits specification of an 'S516 instruction by simply reading or writing the proper address, and permits the use of long-word 68000 instructions to speed up data transfer. The implicit addresses for the 'S516 assume that A1 and A0 (internal) are both zero. Address line A1 will increment during a long-word transfer.

The circuit of Figure 6 is complete (the pull-up resistors are assumed to exist somewhere on the bus). The connection of the error of lowest priority and overflow the next higher priority. One more interrupt line $(\overline{IPL2})$ can be used by another device. The 68000 status register may be set to disable either or both of the 'S516 interrupts.

If more interrupts are required, wire-OR $\overline{ERR1}$ and $\overline{ERR2}$ to provide a single interrupt line, and tie $\overline{IACK1}$ and $\overline{IACK2}$ together. A 74S/LS148/348 priority encoder and 74S/LS138 decoder can be used to accomodate up to eight interrupt and interrupt acknowledge lines, respectively.

Address lines A23 and A22 are decoded to memory-map the 'S516. This permits up to 12 megabytes of system memory without conflict. If more memory is required, use a 74S133 NAND gate, followed by an inverter, to decode A10-A22, and leave EN2 tied to A23. All but 1K bytes of memory space is then available. For full decode of A5-A23, add a 74S30 and another inverter. The 68000 address outputs must be buffered (use 74S244s) to accommodate the added load of the NAND gates.

Bus transceivers can be used (74LS245s) to increase the drive capability of the 'S516 to 24 mA. Tie the 'LS245 enable inputs to GO and direction inputs to R/W. The 'A' I/O lines connect to the 'S516, while the 'B' lines connect to the system bus.

The 74S244, 74LS245 interface circuits and the 74S148, 74S348 priority encoders are available from Monolithic Memories.

Interface Circuit Operation

The timing relationships for the interface are shown in Figure 7. Most of the timing is straightforward, except for the write cycle clock. The 68000 data hold time plus the PAL 16R6A propagation delay time violate the 'S516 data hold requirement. to circumvent this situation, the CK rising edge has been moved to the start of 68000 time slot S7, where setup and hold requirements can be met.

PAL #1 switches CK between the system clock and the read/write strobes. $\overline{GO}$ is edge-triggered by the system clock to provide proper sampling of AS and LDS. In particular, the condition of AS asserted and LDS negated starts the modified write clock. This condition is valid at the start of slot S4, a clock rising edge.

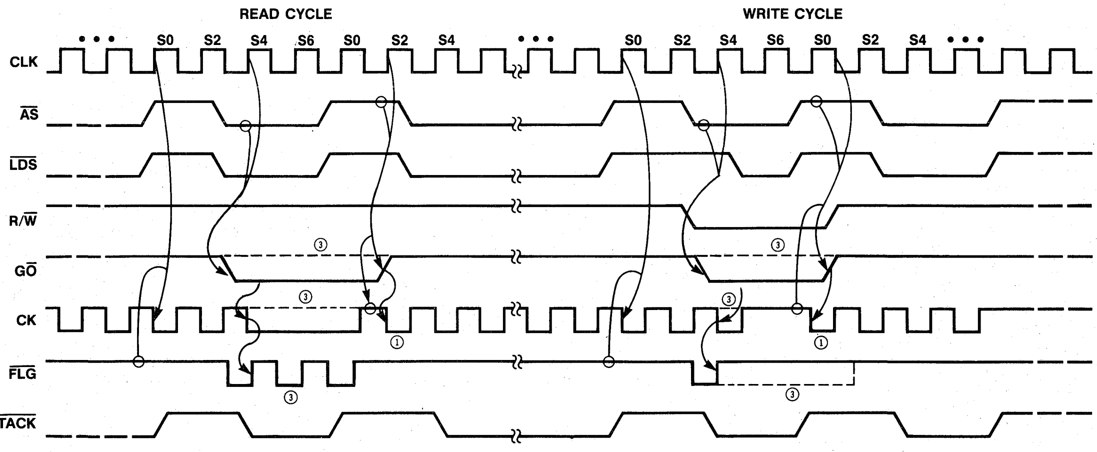
A 5-bit binary counter (QA-QE) serves two functions. During a multiply or divide operation, the counter counts either 8 or 20 clock cycles, respectively, to determine when the 'S516 is done. When done, the counter will hold at count 11000. This count represents states 0, 8, and 10 of the 'S516. Second, loading an input will set QB. Count 11010 thus represents states 1, 3, and 11. QC serves as a guard bit to prevent erroneous 'takeoff' of the counter. When a new multiply or divide is started, the counter is jammed to 10000 or 00100, respectively.

The $\overline{GO}$ and counter terms reject illegal sequences. The 68000 R/$\overline{W}$ line is used to reject instruction 7 write and instruction 0-6 read attempts, with associated bus conflicts. If the counter is not in state 110x0, $\overline{GO}$ is suppressed, but an error condition does not occur.

One 'S516 feature is not supported by PAL #1. Pre-loading of the X register is suppressed, since this merely stops the 'S516 CK (eliminating the benefit of an overlap). A load instruction presented durig the multiply/divide cycle will be delayed until the cycle is completed.

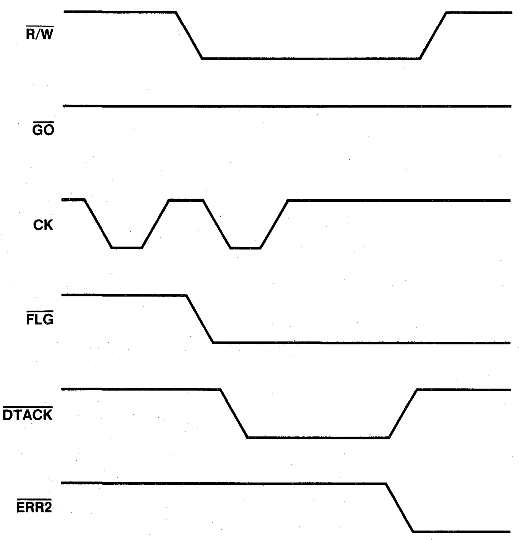
An error condition is indicated by the lock-up of the CK and FLG lines. An asynchronous sequential state machine generates these lines. The next-state table, transition diagram, and Karnaugh maps for this state machine are given in Figures 9, 10, and 11. The state table is free of functional hazards. All terms on the Karnaugh maps overlap, precluding output glitches. The state machine approach ensures reliable control of these two critical outputs.

The design of PAL#2 is straightforward, and requires no further elaboration. The PAL DESIGN SPECIFICATIONs for PAL #1 and PAL #2 are given in Tables 1 and 2.



① Absolute MIN. CK LOW = 68ns AT 6 MHz.
② MIN. CK LOW during WRITE = 68ns at 6 MHz.
③ $\overline{\text{GO}}$, $\overline{\text{FLG}}$ and CK will not toggle upon error, but $\overline{\text{DTACK}}$ will toggle.

Figure 7. Interface Timing



$\overline{\text{ERR2}}$ must be released by accessing the multiplier during the interrupt routine.

Figure 8. Erroneous Write Cycle

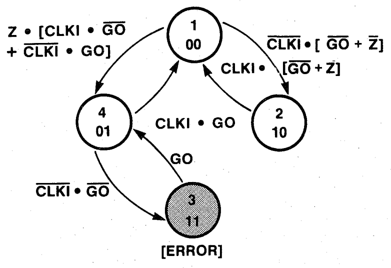


Figure 10. State Transition Diagram for CK-FLG

GO - CLKI - Z

PRESENT STATE	000	001	011	010	110	111	101	100	CK	FLG	
1	2	2	4	1	1	1	4	2	0	0	
2	2	2	1	1	2	1	2	2	1	0	
3	3	3	3	3	4	4	4	4	1	1	[ERROR]
4	3	3	4	4	1	1	4	4	0	1	

$Z' = \text{ENO} * \text{AS} * \overline{\text{LDS}} * \overline{\text{RW}} * \text{QD} * \text{QE} * \overline{\text{GO}}$
$+ \text{ENO} * \text{AS} * \text{RW} * \text{QD} * \text{QE} * \overline{\text{GO}}$
$+ \text{ENO} * \overline{\text{LDS}} * \overline{\text{RW}} * \text{GO}$
$+ \text{AS} * \text{RW} * \text{GO}$

Figure 9. Next State Table for CK-FLG

GO - CLKI - Z

STATE	000	001	011	010	110	111	101	100	CK	FLG	
1	1	1	0	0	0	0	0	1	0	0	
4	1	1	0	0	0	0	0	0	0	1	$\overline{\text{CK}}'$
3	1	1	1	1	0	0	0	0	1	1	
2	1	1	0	0	1	0	1	1	1	0	

1	0	0	1	0	0	0	1	0	0	0	
4	1	1	1	1	0	0	1	1	0	1	FLG′
3	1	1	1	1	1	1	1	1	1	1	
2	0	0	0	0	0	0	0	0	1	0	

GO - CLKI - Z

Figure 11. Karnaugh maps for CK and FLG

Programming Examples

Address definitions

Programming is simplified by first defining absolute short addresses for the 'S516:

```
WMUL0  EQU     $FFE0    ;Instruction 0
WMUL1  EQU     $FFE4    ;Instruction 1
WMUL2  EQU     $FFE8    ;Instruction 2
WMUL3  EQU     $FFEC    ;Instruction 3
WMUL4  EQU     $FFF0    ;Instruction 4
WMUL5  EQU     $FFF4    ;Instruction 5
WMUL6  EQU     $FFF8    ;Instruction 6
RMUL7  EQU     $FFEC    ;Instruction 7
```

The short addresses will be sign-extended to provide the full 24-bit address, i.e. $FFE0 is extended to $FFFFE0. The mnemonics remind the programmer not to write to RMUL7 and not to read WMULx.

The 68000 address registers can also be used, if available. Using the address registers saves the fetch (4 cycles) of the short address during read or write operations.

Examples will use absolute short addresses. Speed comparisons will be stated in clock cycles. Each clock cycle represents 167 nsec at 6 MHz.

'S516 initialization

The gating in PAL #1 precludes the use of the initialization scheme suggested in the 'S516 data sheet. Use the following routine, instead:

```
***    'S516 INITIALIZATION ROUTINE    ***

       MOVE.W  RMUL7.W, D7     ;Reset PAL #1
       MOVE.W  D7, WMUL4.W     ;Start dummy divide
       MOVE.W  RMUL7.W, D7     ;Reset 'S516
```

Multiplication

Multiply routines for the 68000 alone and the 68000/'S516 are now compared. In this benchmark, A0 points to the multiplier, and D0 holds the multiplicand and final 32-bit product.

```
***    68000 MULTIPLY ROUTINE    ***

       MULS     A0@, D0          ;Signed multiply

;      total of 74 cycles


***    68000/'S516 MULTIPLY ROUTINE    ***

       MOVE.W  A0@, WMUL5.W  ;Load X
       MOVE.W  D0, WMUL0.W   ;Load Y
       MOVE.L  RMUL7.W, D0   ;Fetch product

;      total of 44 cycles (0 wait cycles)
```

This is close to a worst-case example. The 1.7 times speed improvement is dictated by the read and write operations, not the 'S516 speed. A faster multiplier could not improve the throughput here, as there are no 68000 wait states introduced (see Figure 12). the 'S516 multiply time is fully imbedded in the 68000 cycles.

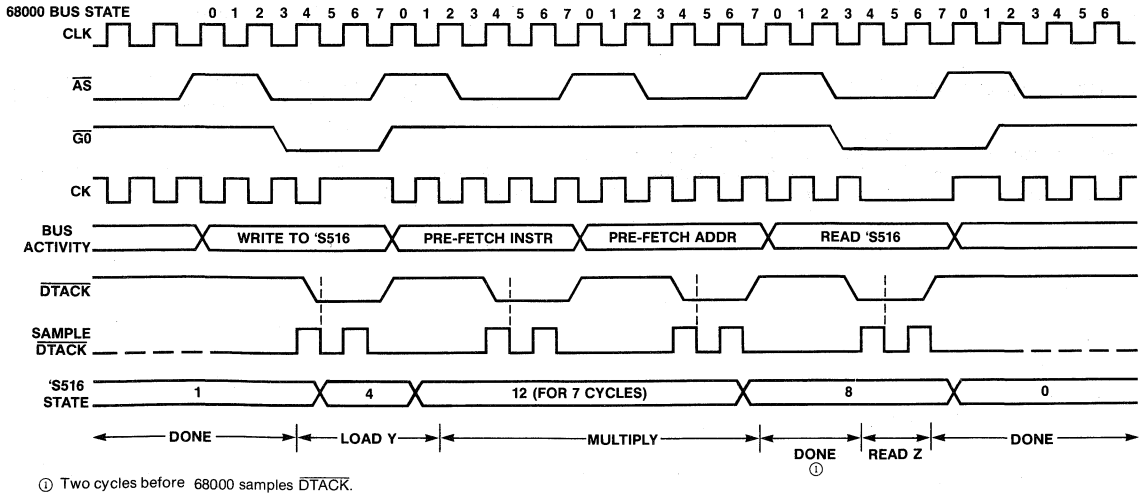
If address registers are available, the routine can be speeded up, as follows:

```
***    68000/'S516 ALTERNATE MULTIPLY ROUTINE    ***

       MOVE.W  A0@, A1@      ;Load X
       MOVE.W  D0, A2@       ;Load Y
       MOVE.L  A3@, D0       ;Fetch product

;      total of 34 cycles (2 wait cycles)
```

Registers A1, A2, and A3 would contain WMUL5, WMUL6, and WMUL7, respectively. The two wait cycles are introduced because the MOVE.L instruction pre-fetch is reduced by 4 cycles, causing $\overline{\text{DTACK}}$ to be asserted 2 cycles after S4 of the MSW fetch. The overall speed improvement is now 2.2 times.



① Two cycles before 68000 samples $\overline{\text{DTACK}}$.

Figure 12. 68000 Instruction Fetch and 'S516 Multiply Imbedded Cycles

More complex operations will show further improvement, as the read/write overhead is reduced relative to the total multiplication time. The improvement will approach the 8 times raw speed difference of the parts. Improvements of more than 8 times are possible for routines that require the chained, fractional, and/or double-precision capabilities of the 'S516.

Division

Let's compare two divide routines, where A0 points to the divisor, and D0 contains the dividend and resulting quotient/remainder:

```
***     68000 DIVIDE ROUTINE    ***

        DIVS      A0@, D0         ;Signed divide

;       total of 162 cycles

***     68000/'S516 DIVIDE ROUTINE    ***

        MOVE.W    A0@, WMUL5.W ;Load X
        SWAP      D0              ;D0 contains MSW
        MOVE.W    D0, WMUL6.W     ;Load Z
        SWAP      D0              ;D0 contains LSW
        MOVE.W    D0, WMUL4.W     ;Load W and go
        MOVE.L    RMUL7.W, D0     ;Fetch result

;       total of 74 cycles (10 wait cycles)
```

The 'S516 provides a 2.2 times speed improvement. Further improvement can be made if address registers can point to the 'S516.

In both the multiply and divide examples fractional arithmetic can be specified by writing X to WMUL6 instead of WMUL5.

Some Final Thoughts

The 68000/74S516 co-processor arrangement has been demonstrated to be practical and effective. Interfacing the two parts requires only two 20-pin PAL devices. Worthwhile performance improvements have been shown.

COST EFFECTIVE!

Acknowledgements

The authors wish to thank Jane Lee and Bill Wickes, whose work served as a foundation for ours.

References

1. C. Hastings, E. Gordon, R. Blasco, *"Minimum Chip-count Number Cruncher Uses Bipolar Co-Processor"*, WESCON/81, Session 3/1

2. E. Gordon, *"A Dedicated Multiplier/Divider Speeds up Multiplication and Division for 8-bit Microprocessors"*, MMI Application Note AN-103.

3. V. Coli, *"SN54/74S08 Interface With the Intel 80855"*, MMI PAL Design Note PD-111/112

4. R. Blasco, *"PROMs, PALs, FIFOs, and Multipliers Team up to Implement Single-Board High-Performance Audio Spectrum Analyzer"*, MMI Application Note AN-100

5. M. Linse, et al, *"The Design and Application of a High-Speed Multiply/Divide Board for the STD Bus"*, NORTHCON/82, Session 15/1

6. T. P. Barnwell, III, et al, *"A Synchronous Multi-Microprocessor System for Implementing Digital Signal Processing Algorithms"*, SOUTHCON/82, Session 21/4

7. J. Birkner, V. Coli, *"PAL Programmable Array Logic Handbook"*, Monolithic Memories, 1981