DEVELOPMENT OF THE

SYMBOLIC PROCESSING ALGORITHMIC RESEARCH COMPUTER

- SPARC -

SEMI-ANNUAL REPORT

REPORTING PERIOD: 78 DECEMBER 01 - 79 JULY 31

Peter G. Juetten, Project Engineer

Phone: 612/853-3713

Contract Effective Date: 79 June 01

Contract Expiration Date: 79 October 30

This research was sponsored by the
Defense Advanced Research Projects
Agency under ARPA Order No.: 3601
Contract No.: MDA903-78-0288
Monitored by: Dr. John Neff, AFOSR/NE

Control Data Corporation
INFORMATION SCIENCES DIVISION
2800 East Old Shakopee Road
Minneapolis, Minnesota 55420

SUMMARY

The SPARC project is a research effort in computer architecture. Initiated by ARPA in 1977, work is being performed by both Carnegie-Mellon University and Control Data Corporation in a joint effort to develop a processor to meet image and signal processing requirements of the 1980's. Utilizing advanced ECL LSI technology, the SPARC processor will initially serve as an add-on device to off-load computational load from a host general-purpose computer.

The goal of this phase of the program is the installation of SPARC processors in laboratories at both CMU and CDC in the late 1979 -- early 1980 timeframe. The hardware is being developed by CDC, along with system software. CMU is developing user software aids, along with a library of applications programs.

At midsummer 1979, the processor design is complete. All parts have been ordered. Cabinet fabrication is in progress, with delivery of the initial cabinet expected in August. The design of an interface between SPARC and the PDP 11 series is complete. Current delivery schedules should allow checkout to begin in the fall of 1979. Initial versions of a cross-assembler and a register-level simulator are complete, along with early diagnostic and system software.

BACKGROUND

Image processing hardware currently available will prove inadequate to meet military needs in the 1980's. The rapidly increasing volume of raw data, combined with increasingly sophisticated processing requirements will exceed present capability by an order of magnitude or more in a few years. In order to satisfy requirements, improvement in both signal-level and symbolic level processing must occur.

Signal-level processing, being generally concerned with arithmetic manipulation of large volumes of data, has been traditionally handled by numeric processors. Increasing volume can be accommodated, to a certain extent, by increasing computational power. To fully meet requirements, however, more processing must be shifted to the symbolic level, which involves non-numeric tasks such as searching and path finding.

PROCESSOR ORGANIZATION

The SPARC processor contains ten functional units connected via a crossbar switch. The operation of these units and the routing of data between them are controlled by microinstructions. Figure 1 shows the SPARC crossbar configuration.

CROSSBAR SWITCH OPERATION

The crossbar switch routes up to 12 16-bit input quantities to a possible 18 destinations on any clock cycle. Figure 2 shows the basic crossbar switch system.

The crossbar switch has three types of conflicts that occur on input to the functional units.

The first type of conflict occurs in a data memory unit when attempting to input the data and the address for the unit simultaneously. The two input lines for the data memory unit are tied together allowing either the data or the address to be input on any cycle. Both the data and address may be input simultaneously if they are the same. The second type of conflict occurs between the control unit and the multiplier unit. The B input to the multiplier unit and the input to the control unit are tied together allowing input to either on any cycle. If both inputs are the same, they may be input simultaneously. The third type of conflict occurs between the shift/boolean unit, the ring port, and the XMAU. The shift count input (C) of BO, the input of XMAUO, and the input to ring port 0 are all tied together. Therefore, any one of the tied inputs may be used on any machine cycle.

MICROMEMORY INSTRUCTION FORMAT

The functional unit operation and data transfer in SPARC are controlled by instructions read from the micromemory. Figure 3 shows the micromemory instruction word format. Bits are numbered from left to right with the most significant (leftmost) bit numbered zero, and the least significant (rightmost) bit numbered 199. All data words in SPARC are numbered in a similar manner.

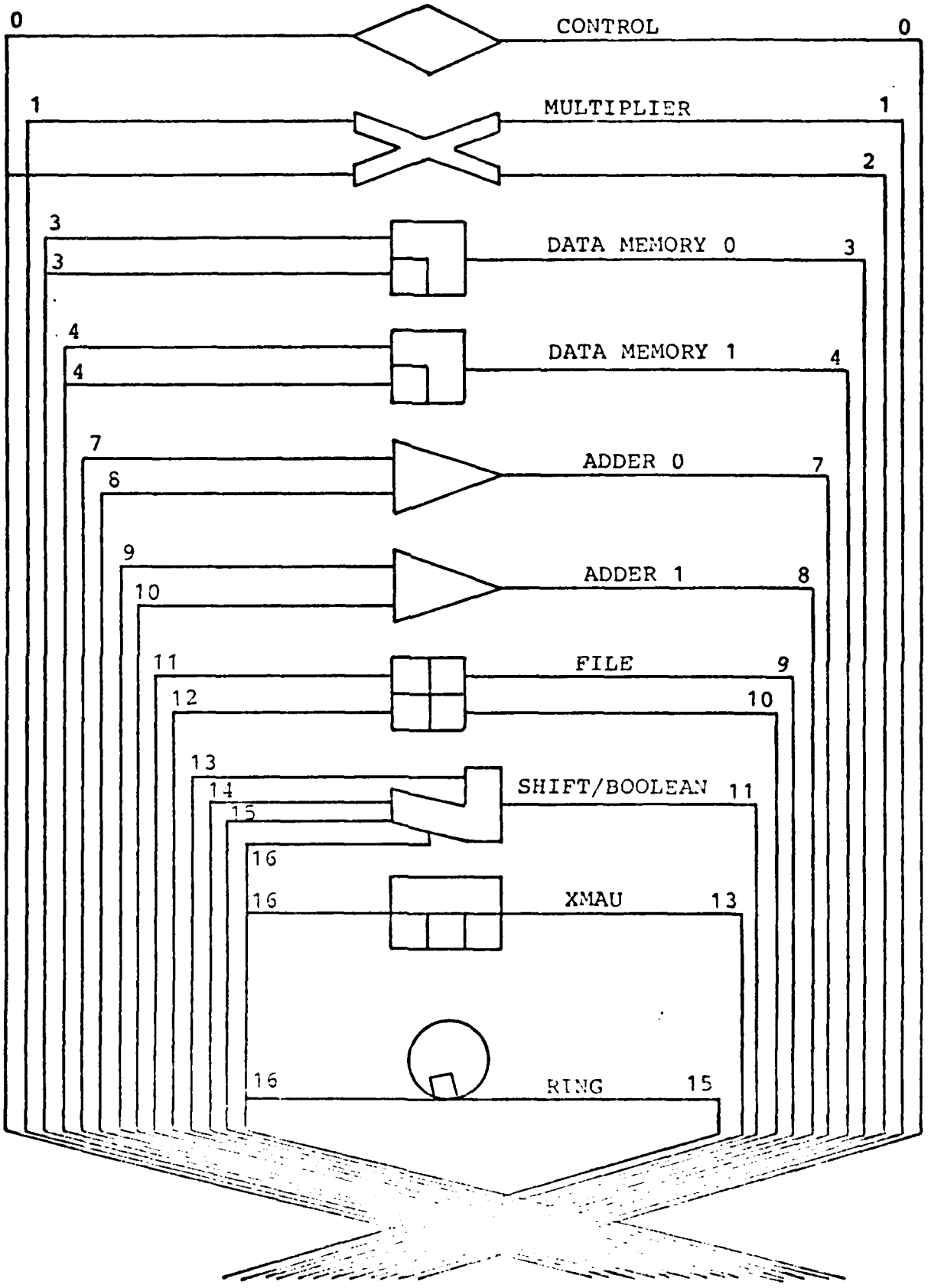

Figure 1. SPARC Crossbar Configuration

J9-1616

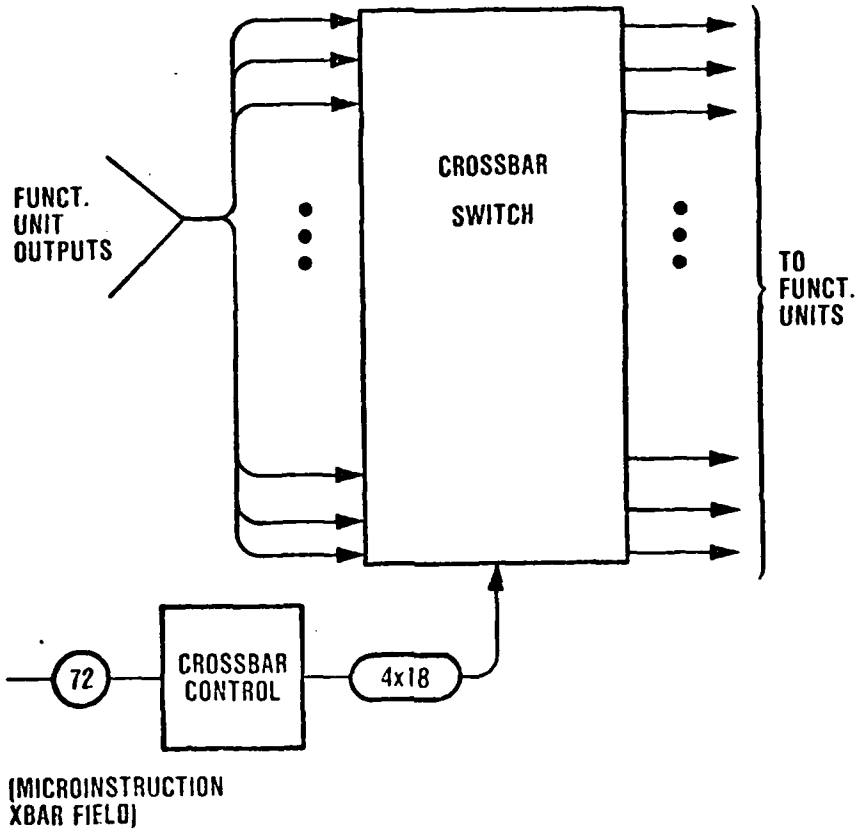

Figure 2. Crossbar Switch System

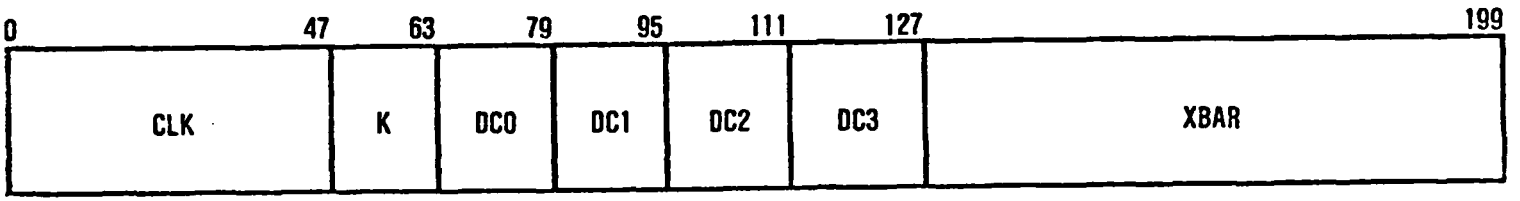

Figure 3. Micromemory Instruction Word Format

Dynamic Control Field (DC)

Each instruction word contains four DC fields. These DC fields carry information transmitted to the specified functional unit where it is latched in an instruction control register and interpreted for the operation to be performed in the unit beginning with the next machine cycle. The information latched is valid until new information is entered. Therefore, it is not necessary to use a DC field each time a result is desired. A DC field is used only to change the unit's operation.

Each DC field is 16 bits wide; the lower four bits specify one of 16 possible functional units and 12 bits are transmitted to the unit specified for interpretation. Any unit may be referenced by any of the DC fields in an instruction. Therefore, the user must be careful not to reference the same functional unit with more than one field in the same instruction (with the exception of the control unit). If this happens, undefined unit operation results.

Constant Field (K)

Each micromemory instruction word contains a 16-bit K field. This field contains a quantity that may be supplied to the crossbar for possible routing to any of the functional units. This field may also be used as a base address for branch instructions.

Clock Control Field (CLK)

Each micromemory instruction word contains a 48-bit CLK field. This field supplies control to each AFP functional unit during each machine cycle. Two or more bits in this field are permanently assigned to each functional unit. In most units, the bits condition the latching of data from the crossbar into the unit's input registers. In certain units (branch unit, file unit, data memories, ring ports, and the XMAUs), some clock bits perform functions not specifically related to clocking of input data. Refer to the functional unit descriptions in this section for an explanation of the clock functions in the individual functional units.

Crossbar Control Field (XBAR)

Each micromemory instruction word contains a 72-bit XBAR field. This Field contains information used to control data routing through the crossbar. The field is divided into 18 4-bit subfields. Each subfield is associated with a crossbar output. The 4-bit code in the subfield specifies which one of 16 crossbar inputs is routed to the crossbar output during the machine cycle.

FUNCTIONAL UNIT DESCRIPTIONS

SPARC contains eight different functional unit types as follows:

- o Data Memory
- o Integer Add
- o Multiplier
- o Shift/Boolean
- o File
- o Ring Port
- o External Memory Access
- o Control

Each unit contains input registers and a control register controlled from the micromemory instruction word and compare hardware enabling a comparison of the result of an operation with a previously defined quantity in the auxiliary register in the unit. The results of these comparisons are available at any time for branching or decision making purposes. For the adder, multiplier, shift/boolean, data memory and file units, an output register is included allowing overlap of transmission of data over the crossbar switch with the next operation within the functional unit specified.

Table 1 specifies the number of cycles needed to complete an operation in each functional unit. These cycle times include the transmission of data over the crossbar switch.

TABLE 1. FUNCTIONAL UNIT CYCLE TIMES

Unit	Number of Cycles For Operation Completion
Data Memory	2 Cycles
Adder	2 Cycles
Multiplier All multiplication operation	3 Cycles
All other operations	2 Cycles
Shift/Boolean	2 Cycles
File Unit	2 Cycles
Ring Port (Direct response from functional unit only)	1 Cycle
External Memory Access (Direct response from functional unit only)	1 Cycle
Control	1 Cycle

Data Memory Unit (D0,D1)

The data memory unit provides storage for variables, intermediate operands, and results during computations. The data memory can also be used as an input buffer for data entering a processor via the ring ports.

There are two data memory units in SPARC, each containing a random access data file with 1024 16-bit words. Each unit also contains a 16-word by 16-bit index file. The index file contains pointers to addresses in the data file, but may be used for additional data storage. The index file entry may be automatically incremented or decremented each memory unit cycle via an adder network after use an indirect address. Comparison networks provide independent testing of the data file address and output.

Each data memory unit receives two inputs from the crossbar switch; one provides direct addressing to either the data file or the index file, and one provides the data to be written. A separate data path, from the I/O section of the processor, enables ring port input data to be written into the data file independently. Direct and indirect addressing is available.

Figure 4 shows the data memory unit. XA provides the address and XB the data. The terms I/O INFO and I/O DATA are the control and data, respectively, of the ring port input data.

The data memory unit has three microinstruction clock bits. Clock A and Clock B control latching of input registers A and B, respectively. Clock C controls all data memory unit operations.

TABLE 2. DATA MEMORY OPERATIONS

Operation	Description
$Z \leftarrow M(A)$	Read data file direct
$Z \leftarrow F(A)$	Read index file direct
$Z \leftarrow F(N)$	Read index immediate
$Z \leftarrow F(N), \wedge F(N)$	Read index, post-increment
$Z \leftarrow F(N), \vee F(N)$	Read index, post-decrement
$Z \leftarrow F(N), I \wedge F(N)$	Read index, post-add
$Z \leftarrow F(N), I \vee F(N)$	Read index, post-subtract

TABLE 2. DATA MEMORY OPERATIONS (Cont'd)

Operation	Description
Z ← M {F(N)}	Read data file indirect
Z ← M {F(N)}, ∧ F(N)	Read data file indirect, post-increment index
Z ← M {F(N)}, ∨ F(N)	Read data file indirect, post-decrement index
Z ← M {F(N)}, I ∧ F(N)	Read data file indirect, post-add index
Z ← M {F(N)}, I ∨ F(N)	Read data file indirect, post-subtract index
M(A) ← B	Write data file direct
F(A) ← B	Write index file direct
F(N) ← B, Z = F(N)	Write index, immediate
M {F(N)} ← B	Write data file indirect
M {F(N)} ← B, ∧ F(N)	Write data file indirect, post-increment index
M {F(N)} ← B, ∨ F(N)	Write data file indirect, post-decrement index
M {F(N)} ← B, I ∧ F(N)	Write data file indirect, post-add index
M {F(N)} ← B, I ∨ F(N)	Write data file indirect, post-subtract index
CA ← XB	Load address comparand register
CD ← XB	Load data comparand register
I ← XB	Load increment register

Eight status bits are generated by each data memory unit. These bits are available as part of the machine condition vector. Two comparisons are made each machine cycle. One operation compares the currently selected data file address, obtained from either input register A or the contents of index file location F(N), and the current contents of the address compare register (CA). The other operation compares the current output of the unit (Z), and the data compare register contents (CD).

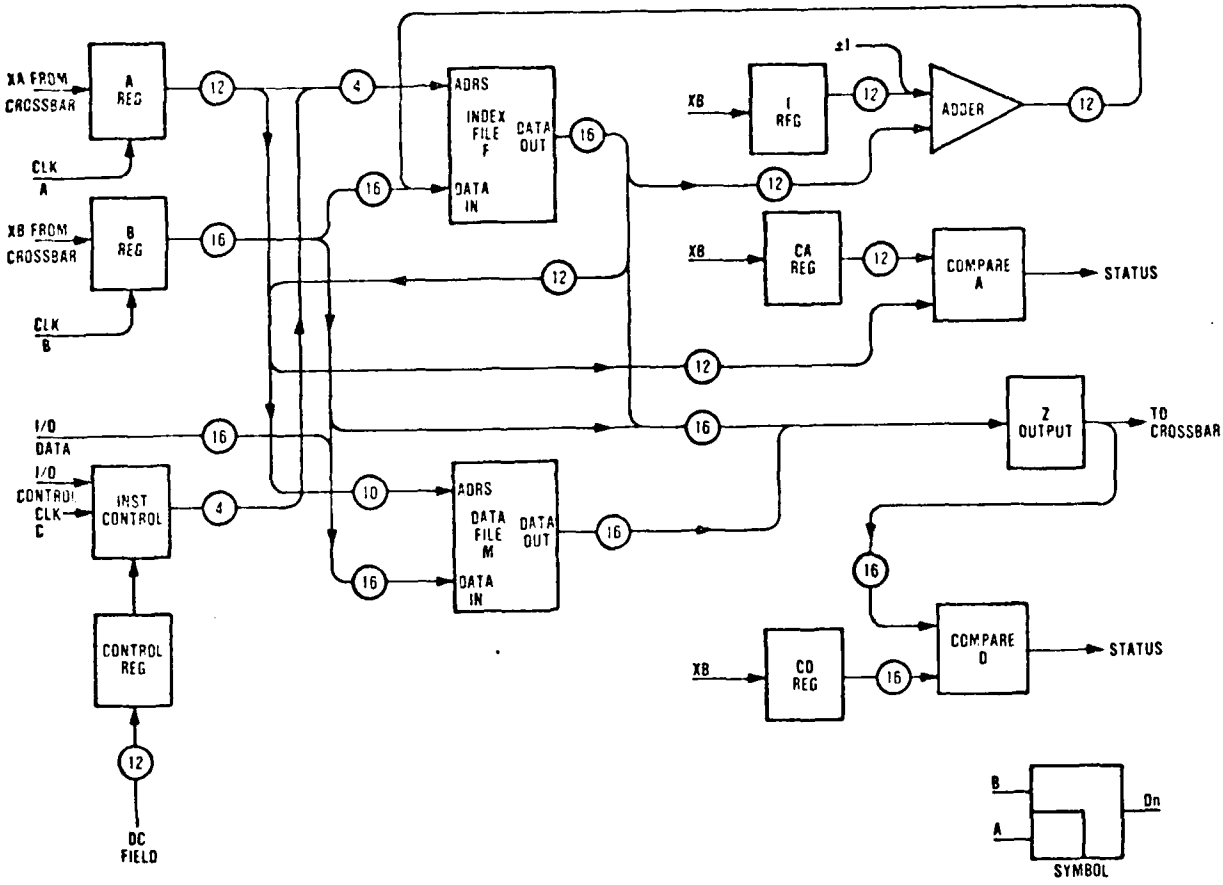

Figure 4. Data Memory Operation

Table 3 summarizes the status bit operations of the data memory unit.

TABLE 3. STATUS BIT OPERATIONS

Operation	Description
A = CA	Address equality
A > CA	Address magnitude
Z = CD	Data equality (16 bits)
ZU = CDU	Upper data equality (8 bits)
ZL = CDL	Lower data equality (8 bits)
ZU $\geq$ CDU	Upper data magnitude (8 bits)
ZU $\geq$ CDL	Lower data magnitude (8 bits)
Z_{MSB}	Most Significant Bit (sign bit)

Integer Add Unit (AO,A1)

The integer add unit forms an integer sum or difference of two 16-bit input operands. The units are capable of performing the following functions.

- o 1's or 2's complement arithmetic operations
- o 8-bit or 16-bit arithmetic operation
- o 32-bit parallel operation
- o Comparison of results
- o Selective latching of inputs A and B

There are two integer add units in SPARC. Figure 5 shows the integer add unit. XA and XB provide the A and B operands from the crossbar.

The add unit has two clock bits. Clocks A and B control the latching of data into the A and B registers, respectively.

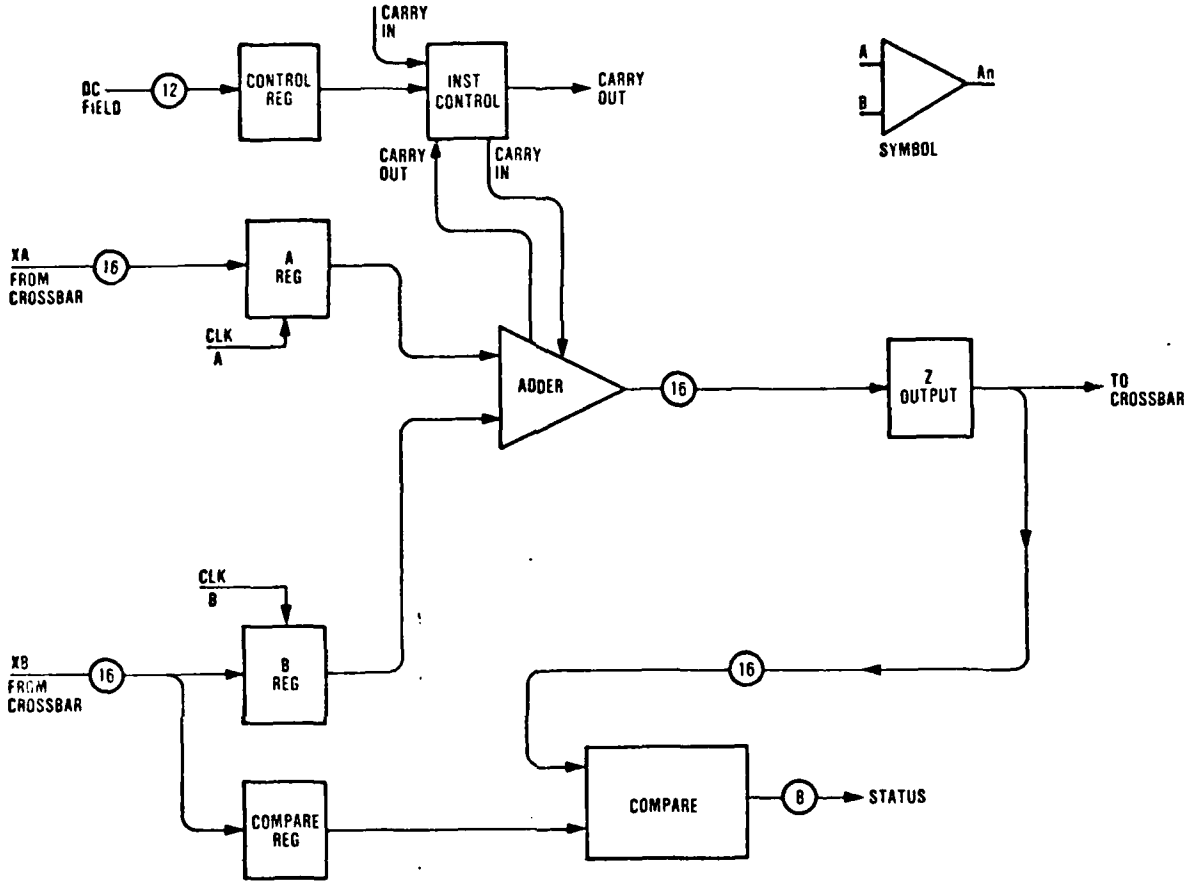

Figure 5. Integer Add Functional Unit

Table 4 describes the functions performed by the integer add unit.

TABLE 4. INTEGER ADD UNIT OPERATIONS

Operation	Description
Arithmetic:	
Z ← A + 0	Arithmetic pass A
Z ← B + 0	Arithmetic pass B
Z ← A + B	Arithmetic add
Z ← A − B	Arithmetic subtract
Z ← B + B	Add B to B
Z ← 0 − B	Minus B
Increment: (2's complement only)	
Z ← A + 1	Add 1 to A
Z ← B + 1	Add 1 to B
Z ← A − 1	Subtract 1 from A
Z ← B − 1	Subtract 1 from B
Z ← A + B + 1	Add 1 to A + B
Z ← B + B + 1	Add 1 to B + B
Z ← A − B − 1	Subtract 1 from A − B
Logical:	
Z ← 0	Word of all zeros
Z ← 1	Word with value of 1
Z ← ONES	Word of all ones
Z ← ⌐ B	Complement of B
Z ← A	Pass A
Z ← B	Pass B
Z ← B + B	Logical left shift 1 (2's complement)
Z ← B + B	Circular left shift 1 (1's complement)
C ← XB	Load comparand register

The integer add unit contains three operation types; arithmetic, increment, and logical. The arithmetic operations operate in 1's or 2's complement mode. The increment operations perform 2's complement arithmetic only. Thus, all 1's complement instructions are mapped to a 2's complement counterpart. The logical operations operate in 1's or 2's complement mode. The default mode for add operation is 2's complement, word (16-bit) operands.

The integer add unit produces 11 status bits defined in Table 5. Two status bits are also generated from combined adder unit operations.

TABLE 5. INTEGER ADD UNIT STATUS BITS

Operation	Description
ZU = CU	Upper data equality (8 bits)
ZL = CL	Lower data equality (8 bits)
ZU > CU	Upper data magnitude (8 bits)
ZL > CL	Lower data magnitude (8 bits)
Z = C	Data equality (16 bits)
Z > C	Data magnitude (16 bits)
Z_{MSB}	Most significant bit of output (sign bit)
ZU = 0	Upper 8 bits zero
ZL = 0	Lower 8 bits zero
A_{CY}	Carry
A_{OV}	Overflow
Z01 > C	32-bit data magnitude (both adders combined)
Z01 = C	32-bit data equality (both adders combined)

Multiply Unit (MO)

The multiply unit performs integer multiplications, population counts, significance counts, and bit reversals. The resulting product of a multiplication operand can be 32 bits (16 bit operands) or two 16-bit products (8-bit operands).

The unit is segmented allowing new operands and new results to be obtained every cycle.

Figure 6 shows the multiply unit. There is one multiply unit in each AFP. The unit has two 16-bit inputs from the crossbar, XA and XB, and two associated clocks A and B, which control latching of data into the respective input registers.

Table 6 describes the operations performed by the multiply unit.

TABLE 6. MULTIPLY UNIT OPERATIONS

Operation	Description
HL ← A*B	16-bit 2's complement multiply
HL ← A*B	16-bit magnitude multiply
H←A, H←B	Pass A and B
H ← (AU*BU), L ← (AL*BL)	
H ← (AU*BL), L ← (AL*BU)	
H ← (AL*BU), L ← (AU*BL)	8-bit multiply
H ← (AL*BL), L ← (AU*BU)	
H ← PC(A), L ← PC(B)	Population counts of A and B
H ← RV(A), L ← SC(B)	Bit reversal of A and significance count of B
H ← PC(A), L ← B	Population count of A and pass B
H ← PC(A), L ← SC(B)	Population count of A and significance count of B
H ← A, L ← PC(B)	Pass A and population count of B
H ← A, L ← SC(B)	Pass A and significance count of B
H ← RV(A), L ← B	Bit reversal of A and pass B
H ← RV(A), L ← PC(B)	Bit reversal of A and population count of B

TABLE 6. MULTIPLY UNIT OPERATIONS (Cont'd)

Operation	Description
CL ← XB	Load lower comparand register CL
CH ← XA	Load high comparand register CH
CL ← XB, CH ← XA	Load comparand registers CL and CH

8-Bit Magnitude Multiply

This operation forms two independent 16-bit products simultaneously from various combinations of 8-bit halfwords from the input registers. Each input register contains two independent halfwords, the upper halfword (most significant 8 bits) designated AU and BU, and the lower halfword (least significant 8 bits) designated AL and BL. The halfwords are considered positive 8-bit integers.

Population Count

The population count is a binary count of the number of 1 bits in an input word. The population count results are stored in the least significant five bits of the multiply outputs in the form of binary integers. The A register count goes to the H output and the B register count goes to the L output. On any cycle, the count of A, B, or both A and B may be made.

Bit Reversal

The bit reversal operation is defined as follows:

$$r_{n-i} = x_i \qquad \text{for } i = 0,1,\ldots,n$$

where r = reversal bit
x = input bit
n = word length - 1

This function is implemented for the A register only with the result going to the H output.

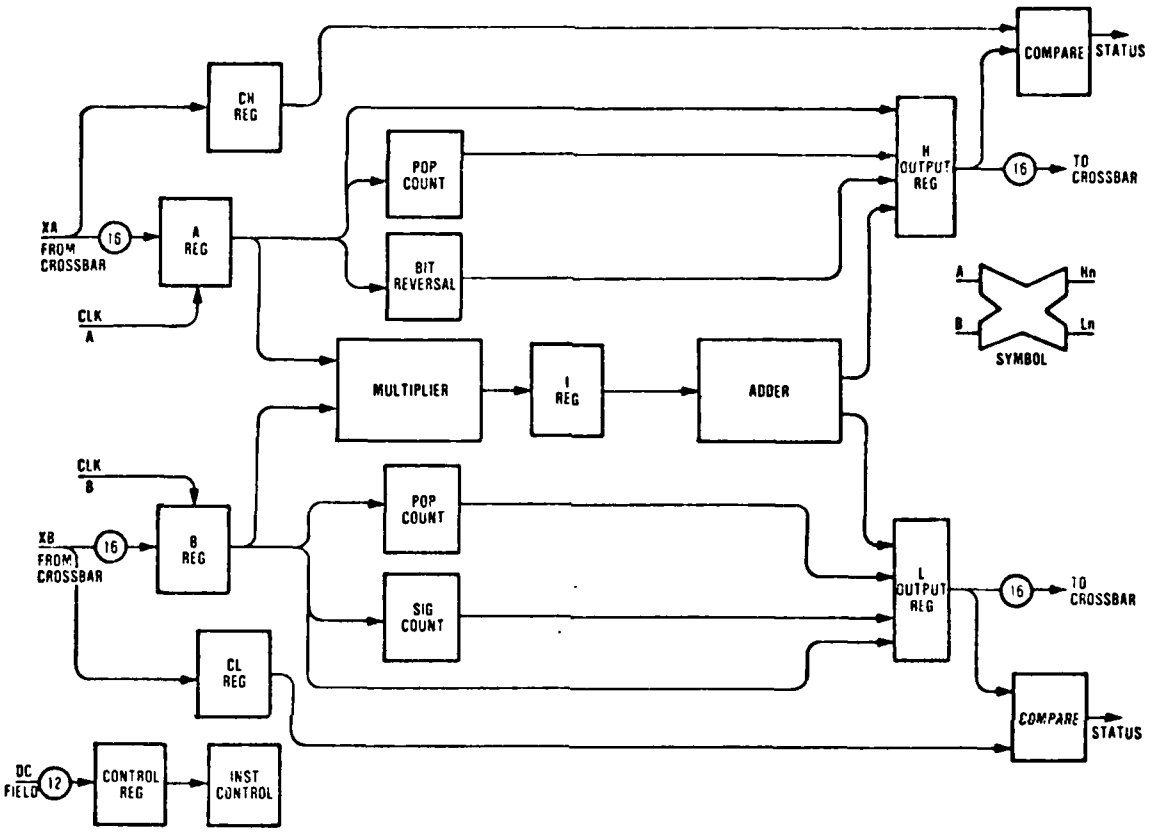

Figure 6. Multiply Unit

Significance Count

The significance count operation identifies the position of the most significant 1 bit in the input word. The result is a positive binary integer, identifying the displacement from the right (least significant) position in the word. The result is contained in the lower four bits of the output, except for an input word of all zeros. For this word, the result contains a 1 in bit position 11 and all other bit positions zero (value of 16).

The significance count operation is implemented for the B register only with the result going to the L output.

Status Bits

The multiply unit produces five status bits defined in Table 7.

TABLE 7. MULTIPLY UNIT STATUS BIT OPERATIONS

Operation	Description
H = CH	H output equality (16 bits)
L = CL	L output equality (16 bits)
H > CH	H output greater than CH based on 16-bit magnitude comparison
L > CL	L output greater than CL based on 16-bit magnitude comparison
H_{MSB}	The most significant bit (bit 0) of the H output
L_{MSB}	The most significant bit (bit 0) of the L output.

Shift/Boolean Unit (BO)

The shift/boolean unit performs shift and bit logical operations. Barrel shifts of 0 to 15 places are performed on 16-bit single operands or 32-bit double word operands. This feature is useful for field extraction over word boundaries.

The unit performs right shifts only. Sign extend, zero fill, and circular (end around) shifts are available. The shift count may be obtained from either the microinstruction control field or other SPARC units via the crossbar switch.

The boolean section performs all 16 logical functions of two variables. The operands used are the output of the shift network (A) and the contents of an input register (B) loaded from the crossbar.

There are four clock bits associated with the shift/boolean unit. Clocks A and D control the latching of data into the shift input registers S and D, respectively, where D is the more significant word in a double word shift operation. Clock B controls the latching of data into boolean input register B. Clock C controls the latching of data into the shift count register C.

Figure 7 shows shift/boolean unit.

Table 8 lists the operations performed by this unit.

TABLE 8. SHIFT/BOOLEAN UNIT OPERATIONS

Operation	Description
Shift K, zero fill	Shift contents of S register right K places, enter zeros from left
Shift K, sign extend	Shift contents of S register right K places, copy bit 0 from left
Shift K, circular	Rotate the contents of S register K places right, end around shift
Double shift K	Shift contents of S register and D register right K places
$Z \leftarrow A \wedge B$	A AND B
$Z \leftarrow A \wedge \neg B$	A AND NOT B (Inhibit)
$Z \leftarrow A$	Pass A

TABLE 8. SHIFT/BOOLEAN UNIT OPERATIONS (Cont'd)

Operation	Description
$Z \leftarrow A \wedge B$	NOT A AND B (Inhibit)
$Z \leftarrow B$	Pass B
$Z \leftarrow A \oplus B$	Exclusive OR
$Z \leftarrow A \vee B$	A OR B
$Z \leftarrow A \downarrow B$	NOT (A OR B) (NOR)
$Z \leftarrow A \equiv B$	Equivalence
$Z \leftarrow \neg B$	NOT B
$Z \leftarrow A \vee \neg B$	A OR NOT B (IMPLICATION)
$Z \leftarrow \neg A$	NOT A
$Z \leftarrow 0$	All Zeros
$Z \leftarrow \neg A \vee B$	NOT A OR B (Implication)
$Z \leftarrow 1$	All ones
$Z \leftarrow A \uparrow B$	NOT (A AND B) (NAND)
$C \leftarrow XB$	Load comparand register

The shift/boolean unit generates seven status bits from the compare network.

The quantities compared are assumed to be positive binary integers. Table 9 lists the status bits and descriptions.

TABLE 9. SHIFT/BOOLEAN UNIT STATUS BITS

Status Bit(s)	Description
ZU = CU	Upper halfwords of C and output are identical
ZL = CL	Lower halfwords of C and output are identical
ZU > CU	Upper halfword of output is greater than upper halfword of C
ZL > CL	Lower halfword of output is greater than lower halfword of C

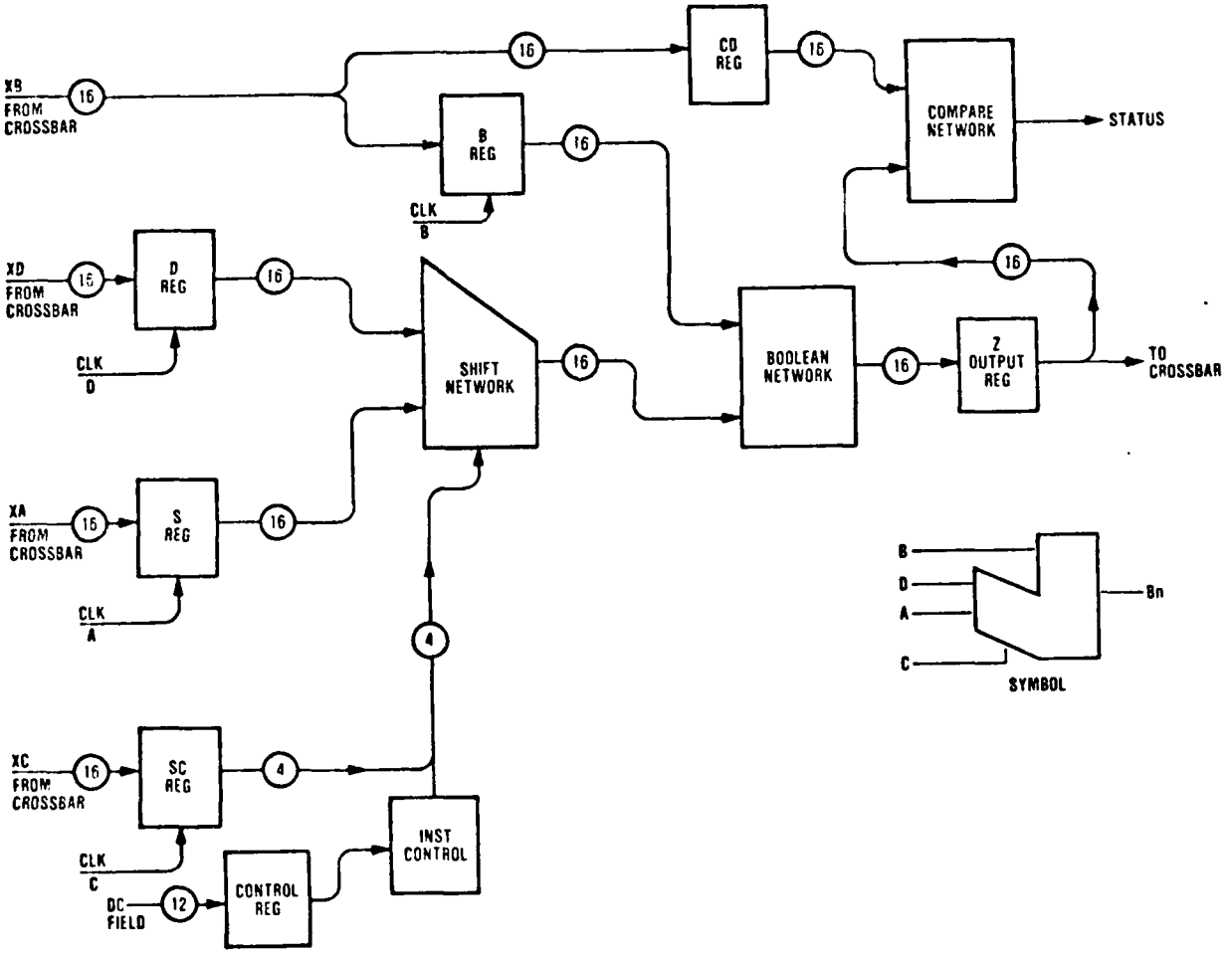

Figure 7. Shift/Boolean Unit

TABLE 9. SHIFT/BOOLEAN UNIT STATUS BITS (Cont'd)

Status Bit(s)	Description
Z = C	The two 16-bit quantities are identical
Z > C	The 16-bit output (Z) is greater than the contents of register C
Z_{MSB}	The most significant bit of the output

File Unit (FG)

The file unit consists of two general-purpose register files. These files provide temporary storage for operands and may be used as delays to coordinate timing between data paths.

The two halves of a file unit operate independently. Each is assigned a separate crossbar input and output, and is supplied with independent read and write addresses obtained from the same dynamic control field. Each half performs a write followed by a read each machine cycle.

Each half of a file unit contains storage for 16 16-bit data words of which only eight words are accessible during only one cycle.

SPARC contains one file unit. Figure 8 shows the file unit. The file unit halves are referred to as file F and file G. They both work in the same manner.

The write data operation occurs unconditionally every time the input register contents are changed. Therefore, previously stored data at the write address is lost unless the write address is also changed in the DC field.

There are three clocks in the file unit. Clocks A and B latch the crossbar inputs XA and XB into registers A and B, respectively. Clock C provides the fourth address bit for the read and write portion of both halves; therefore, selecting the upper or lower eight words in each half of the unit.

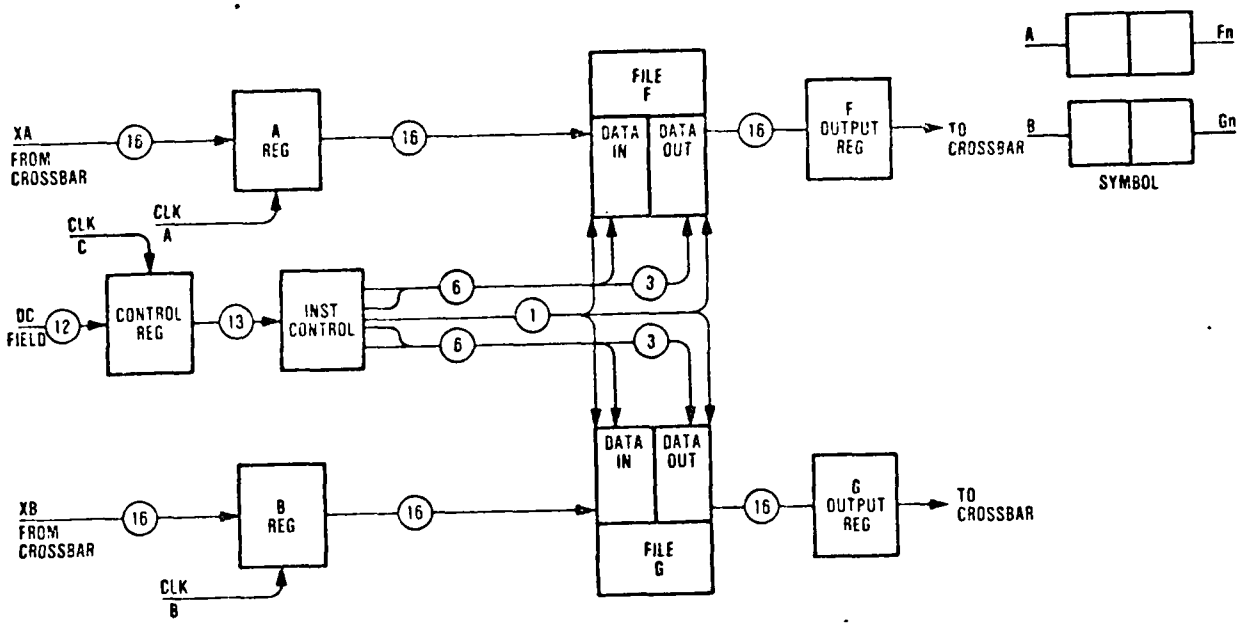

Figure 8. File Unit

There are two status bits available from the file unit as follows:

F_{MSB}	The most significant bit of the F output
G_{MSB}	The most significant bit of the G output

Ring Port (IO)

The ring port provides an interface between SPARC and an interprocessor communication ring. All information arriving at the processor's ring port input register is decoded. Information intended for the processor is sent to its destination. Other ring packets are transmitted to the next processor in the ring. A ring packet consists of one 16-bit data word, eight bits of addressing information, and four control bits. Data on the ring has priority over data entering the ring from the output first in first out (FIFO) buffer.

In addition to data transfer and synchronization functions, the ring ports provide the primary external access to and control over the processor. Only one ring port (I/O) in a processor accepts control information. Figure 9 shows the ring port unit.

The ring port unit has two clocks. Clock A executes the current output control operation. Clock B clears the status register bits corresponding to the mask register bits.

Output Section

The output section of the ring port contains a 16-entry, FIFO buffer between the ring port crossbar input register (A) and the ring output register. Therefore, the processor can write up to 16 words to the ring port, even when the ring is stopped or full, before a processor stop condition is generated. This stop condition occurs only if the processor attempts to write to the ring when the output FIFO is full and is released when the output FIFO is capable of accepting another word of output.

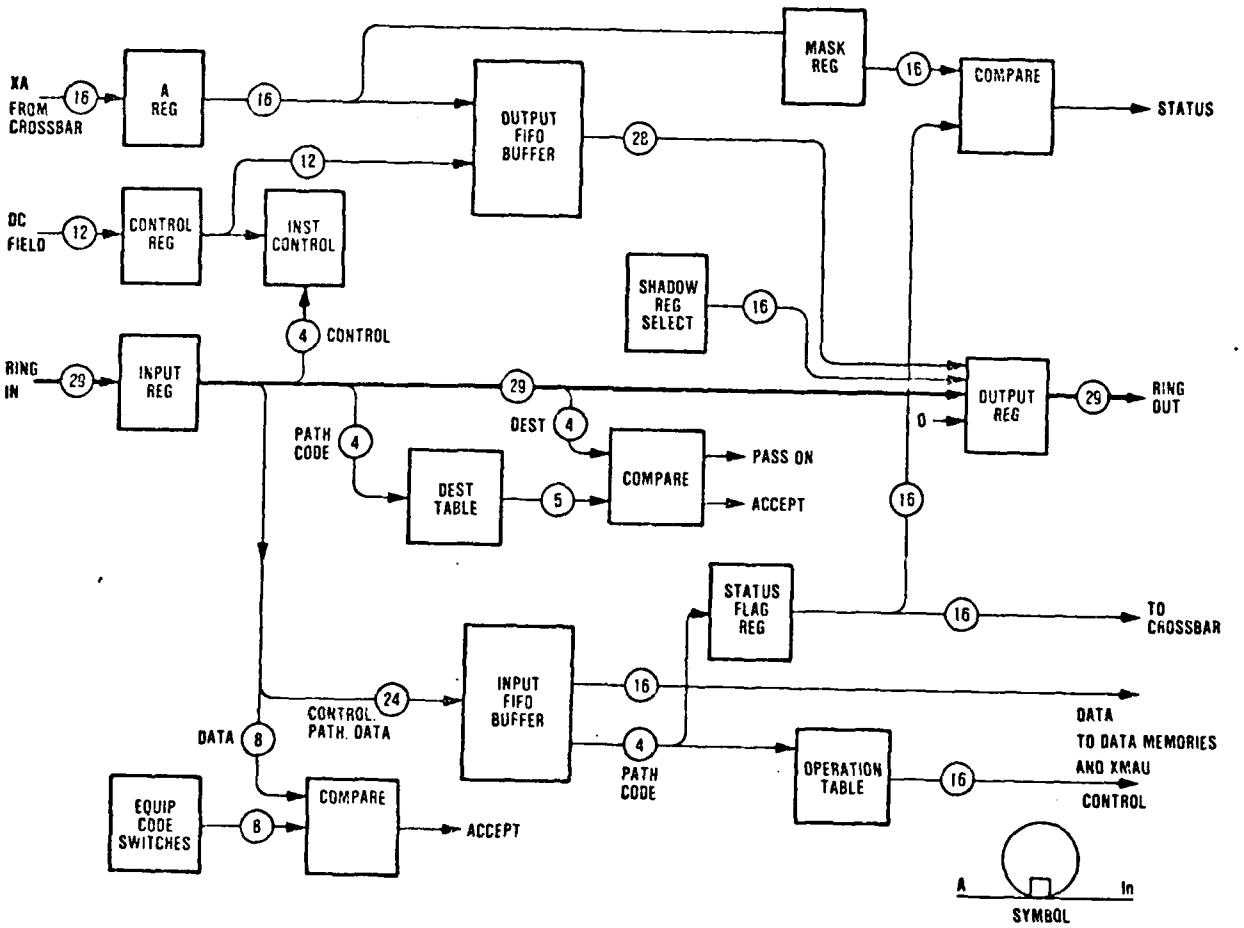

Figure 9. Ring Port Block Diagram

Ring Section

The ring section contains the input and output transfer registers of the ring. Ring packets may be clocked from a previous port on the ring into the input register, from the input to the output register, and from the output register to the next port every machine cycle. Along with each packet is a bit indicating whether the packet has been around the ring more than once. This bit is set when the packet passes a specific point on the ring, normally the interface or controlling processor, and signals an error condition if it reaches the same point a second time.

Input Section

The input section of the ring port contains the processor addressing logic, buffering of incoming data and control packets, and automatic distribution of data into various memories of the processor.

There are two modes of processor addressing depending on a control bit in the ring packet; direct and indirect.

For forced transfer packets, direct addressing is used. The lower eight bits of data in the ring packet are compared with the eight equipment code switches and if a match occurs, the packet is accepted. The remaining parts of the input section are bypassed and the packet is acted upon directly.

For indirect addressing, a 4-bit path code in the ring packet is used to address one of the 16 destination table entries. The 4-bit destination field in the ring packet is compared with four bits of the table entry and if a match occurs the packet is accepted. The fifth bit of the destination table entry determines whether the packet is removed from the ring or allowed to continue around to another processor. This allows data and/or synchronization signals to be broadcast to a number of processors simultaneously.

The input section contains a FIFO buffer of 16-word by 24-bits where data, the path code, and control bits of an incoming indirectly addressed packet are stored. The packet is processed only when leaving the input FIFO.

The operation table contains 16 entries addressed by the path code. An entry points to the data memory where the data word is to be written, identifies an index register in that memory to be used for the address, its post increment mode, and an associated status flag. Therefore, up to sixteen buffers each with an associated status flag may be defined in any of the data memories for each ring port.

The status flag register has an associated 16-bit mask register which may be set by the receiving processor. Two status conditions are generated using this mask register; the logical AND may be used to coordinate the arrival of data from different sources with a single test. The logical OR could signal any one of a number of inputs available for processing. There is a processor instruction (clock bit) that clears all status flags with a corresponding mask bit set. This instruction generates a processor stop condition if any selected bit is not currently set. This ensures that the processor waits for all incoming data before processing it.

The set status signals can be sent with the last word of data to signal arrival of a buffer, with the first word of data to prevent the data from overwriting the buffer before it is available, or by themselves for synchronizing processes or signaling special events.

Status Bits

The ring port produces four status bits which indicate whether the output FIFO buffer is full, any status-mask pair is set, all status-mask pairs are set, or the ring is stopped.

External Memory Access Unit (XO)

The external memory access unit (XMAU) provides the facility for transfer of information between SPARC and system bulk memory. Bulk memory banks process data in blocks of four 16-bit words called super words (swords). Figure 10 shows the XMAU.

The XMAU contains a four-word assembly/disassembly data register to hold the sword. One 16-bit word may be read and/or loaded via the crossbar each cycle. The word is defined by an assembly/disassembly (AD) counter (two bits), loaded from the instruction control. Communication with the bulk memory system is performed via a 64-bit wide data path with some buffering to allow for a read lookahead capability. The XMAU contains two address registers (16-bit), 1 and 2, each loaded directly from the crossbar. A segment register also loaded directly from the crossbar allows addressing of up to 2^{32} swords of memory.

The two input buffers each hold up to eight swords of data. The 64 bits of input from the bulk memory to the input buffer are loaded into the normal or interrupt sections depending on the mode of the processor when the read request was executed.

The XMAU is capable of transferring to or from the crossbar one 16-bit word every machine cycle when accessing words sequentially without a break at a sword boundary provided no memory bank conflicts with other XMAU's occur.

Swords may also be requested in a random manner with the address provided explicitly. Requests may be made for lookahead purposes before any data is processed.

The SPARC system, as initially configured, will contain no bulk memory. A single XMAU will be included, however, because it is required for loading and examination of micromemory. The presence of this unit will facilitate later system expansion via the addition of bulk memory.

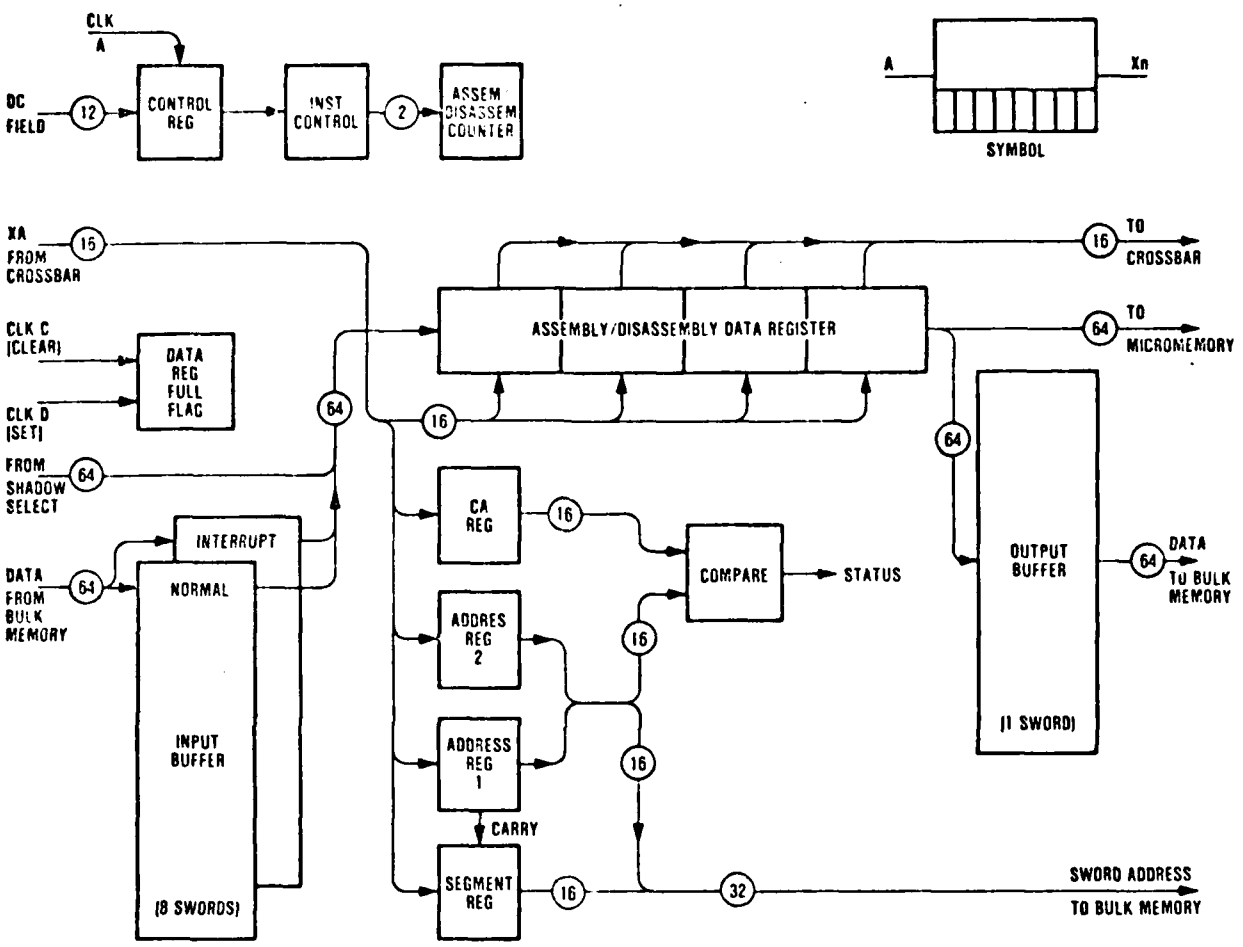

Figure 10. External Memory Access Unit

Write requests may be interleaved with read requests.

A direct path is provided to micromemory. This path is 64 bits wide and facilitates rapid loading of overlays into micromemory. The micromemory contents may also be modified and/or examined by the processor via this path. Table 10 lists the operations performed by the XMAU.

TABLE 10. EXTERNAL MEMORY ACCESS UNIT

Operation	Description
Group 1 (Bulk Memory Access)	
DR ← BM(SG.A1)	Read bulk memory random - register 1
BM(SG.A1) ← DR	Write bulk memory random - register 1
DR ← BM(SG.A1)	Read bulk memory sequentially - register 1
SG.A1 ← SG.A1+1	
BM(SG.A1) ← DR	Write bulk memory sequentially - register 1
SG.A1 ← SG.A1+1	
DR ← BM(SG.A2)	Read bulk memory random - register 2
BM(SG.A2) ← DR	Write bulk memory random - register 2
DR ← BM(SG.A2)	Read bulk memory sequentially - register 2
A2 = A2+1	
BM(SG.A2) ← DR	Write bulk memory sequentially - register 2
A2 ← A2+1	
Group 2 (XMAU Register Operations)	
DR(AD) ← XA	Load data register from crossbar (write 16 bits)
A1 ← XA	Load address register 1
A2 ← XA	Load address register 2
SG ← XA	Load segment register
CA ← XA	Load address comparand register
SG.A1 ← SG.A1+1	Increment address register 1
A2 ← A2+1	Increment address register 2

TABLE 10. EXTERNAL MEMORY ACCESS UNIT (Cont'd)

Operation	Description
Group 3 (DR Access Mode)	
AD ← 0	Clear assembly/disassembly counter
AD ← 1	Load assembly/disassembly counter with value 1
AD ← 2	Load assembly/disassembly counter with value 2
AD ← 3	Load assembly/disassembly counter with value 3
AD ← AD-1	Post-decrement assembly/disassembly counter
AD ← AD+1	Post-increment assembly/disassembly counter
DR ← MM	Read micromemory shadow register
MM0 ← DR	Write micromemory upper - parcel 0†
MM1 ← DR	Write micromemory lower - parcel 1†
MM2 ← DR	Write micromemory upper - parcel 2†
MM3 ← DR	Write micromemory lower - parcel 3†
Z = DR(AD)	Data register output (available at all times)

† These instructions are available only on the master XMAU (X0)

The bulk memory may be accessed for read or write in two modes; random or sequential.

In random mode (sword mode), every time the unit is executed (clock A set), a request (read or write) is sent to the bulk memory with the contents of the specified address register. In the case of a write, the contents of the data register (DR) is also sent. In both cases, the new contents are used if the same instruction loads the address or data register. This mode is

normally used only when every execution of the unit requires a new sword (64 bits), such as prefetching the first few swords of an array or only one word is to be accessed in each sword.

In sequential mode (word mode), bulk memory references are made automatically on sword boundaries. Every time the AD counter wraps around (that is, increments from 3 to 0 or decrements from 0 to 3) a bulk memory reference is initiated at the current (or new address if loaded in the same instruction), the address is post-incremented and the data register full (DR full) flag is cleared. The result of clearing DR full is the next (prefetched) sword waiting in the XMAU's input buffer is transferred into the DR and the DR full flag is reset. This operation occurs such that with adequate prefetch and no serious bank conflicts data can be read at the rate of one word per machine cycle without a break at sword boundaries. If the selected address is greater than the address comparand, bulk memory references are not made. This can be used to prevent unnecessary prefetches at the end of an array. Normally, a maximum of eight sword prefetches may be issued before any of the data is used.

The XMAU registers may be manipulated at the same time bulk memory references are being made. The address registers may be loaded while in sequential mode. This pseudo random access is the normal method of performing random addressing of bulk memory when all words within the swords are being used. Explicitly incrementing the address registers in sequential mode is not required and can produce unpredictable results.

The AD counter can be manipulated, the micromemory written, or the shadow register network read simultaneously with the above operations. In sequential mode, the AD counter must be specified as increment or decrement for wraparound to occur.

NOTE

The bulk memory access mode (including the wraparound condition) must not be changed in the same cycle in which the unit is executed and such execution would result in a bulk memory reference. This constraint does not apply to reads which have six or less prefetches which have not yet been used.

The XMAU has four clock bits as follows:

Clock	Description
A	Execute current XMAU operation.
B	Wait for data ready bit- an instruction is reading the data register and stops the processor until the data register is full
C	Clear data register full flag- processor has used data register data
D	Set data register full flag- prevents loading prefetched data over current DR contents

There are seven status bits produced by the XMAU. Table 11 lists these status bits.

TABLE 11. XMAU STATUS BITS

Operation	Description
A = CA	Address register equivalence
A > CA	Address register greater than comparand
U_{MSB}	Most significant bit upper byte
L_{MSB}	Most significant bit lower byte
Data Register Full	Status check
Output Buffer Full	Status check
Request Stack Full	Status check

Control Unit (CU)

Control of functional unit operation and data transfers in SPARC is provided by machine instructions stored in micromemory. This memory operates independently of the previously described data memories. The instructions read from this memory contain information which designates functional unit operations, controls data passing into the functional units, routes data between the units, detects conditions arising from within the processor and external to it along with the ability to appropriately modify machine control as a result of these conditions, and directs data into and out of the processor via the various input/output methods previously described. Figure 11 shows the control unit.

The control section consists of the following components:

- o Micromemory
- o Crossbar switch
- o Branch logic
- o Interrupt network

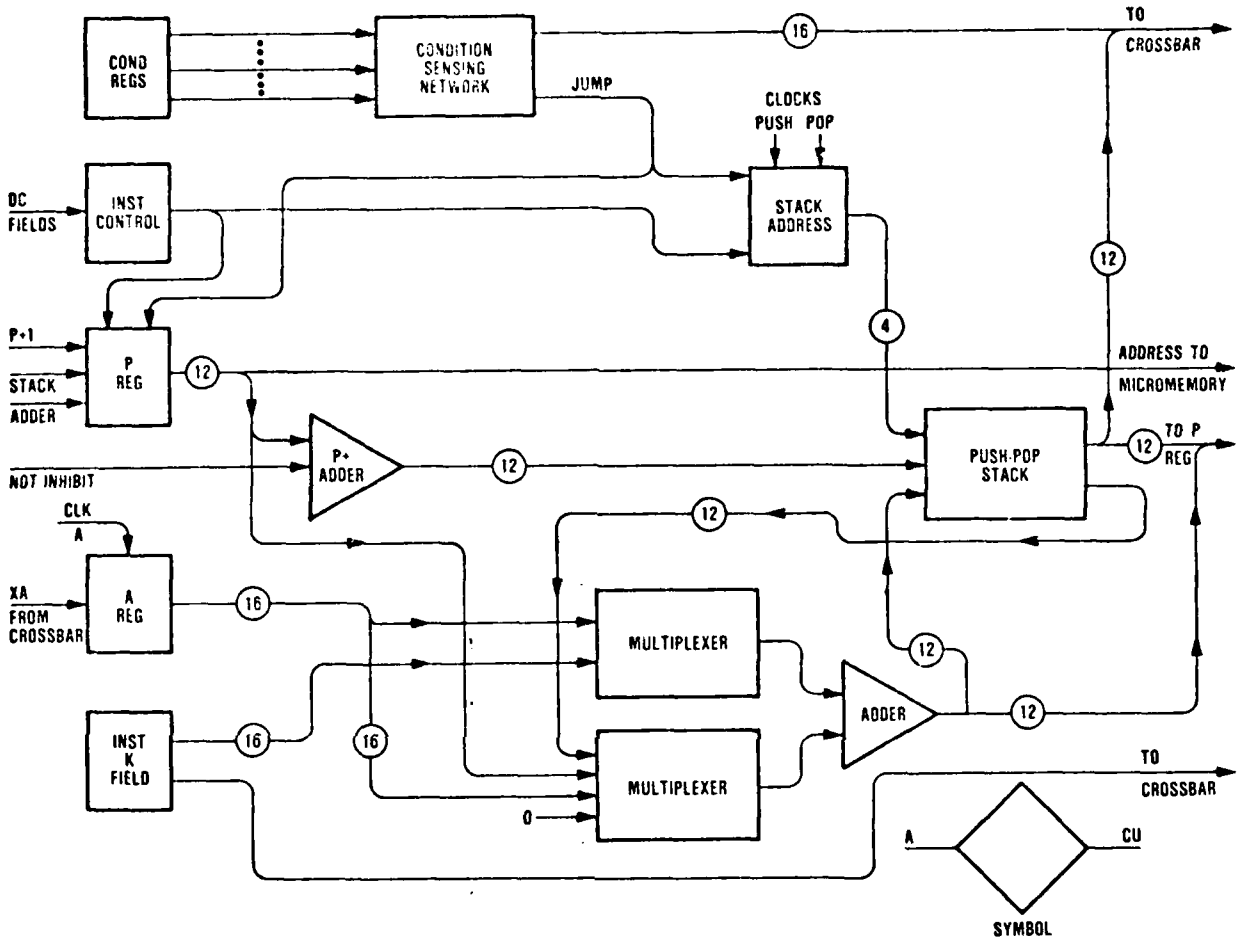

Figure 11. Control Unit

Micromemory

The micromemory in SPARC holds all the instructions used to control machine operations. The micromemory contains 1024 words each 200 bits in length. It is capable of being loaded either via forced transfers from the master ring port or under program control directly from an external memory access unit. Refer to Micromemory Instruction format in this section for further information.

Crossbar Switch

The crossbar switch provides data paths between the outputs and inputs of the functional units (refer to Figure 1).

The switch consists of an 18 input, 16 output crosspoint mechanism. Each path through the switch is 16 bits wide. All 16-bit input quantities may be simultaneously routed to any of the output ports in a machine cycle. Each output port drives up to four functional units. Refer to Crossbar Switch Operation in this section for further information.

Branch Logic

At the end of each machine cycle the conditions of various functional units become valid and are latched into condition registers in the units. These conditions may be sensed by selecting the control unit using a DC field of the microinstruction. A conditional branch may be selected in each of the four DC fields allowing up to four conditions to be sensed during the same instruction. The branch address should be the same for all conditions selected for proper operation. The branch address is taken if any of the selected conditions is true.

P Register

SPARC control contains the 12-bit P register. This register contains the address of the next instruction word being fetched from the micromemory. An increment network advances the contents of the P register by one during normal sequential program execution. Other inputs may be latched in the P register during the execution of branch instructions.

P Stack

A 16-word pushdown stack is associated with the P register. This stack is used to store program addresses for later recall. The stack may be pushed, popped, written, or read under program control.

Control Unit Operations

Table 10 lists the operations performed by the control unit. The control unit has five clock bits. Clock A latches data into input register A, clock B increments the P stack, clock C decrements the P stack, clock D inhibits execution of the next instruction on jumps, and clock E inhibits execution of the next instruction fall-through.

TABLE 10. BRANCH UNIT OPERATIONS

Operation	Description
P ← S	Jump to location contained in stack
P ← P + K	Jump to current location + constant
P ← K + A	Jump to constant + crossbar input
P ← K	Jump to constant
P ← S + A	Jump to stack + crossbar input
P ← P + A	Jump to location of next instruction + crossbar input
P ← A + A	Jump to crossbar input doubled
P ← A	Jump to crossbar input
S_C ← 0	Stack address clear
S ← P + K	Stack location of next instruction + constant
S ← K + A	Stack constant + crossbar input
S ← K	Stack constant
S ← S + A	Stack current value + crossbar input
S ← P + A	Stack current location + crossbar input
S ← A + A	stack crossbar input doubled
S ← A	Stack crossbar input

Interrupt Network

A single level interrupt system is implemented in SPARC. Interrupts may be initiated from any of the ring ports included in a processor, and may be selectively enabled or disabled from each individual port. The interrupt system may also be activated or deactivated. The machine state, as represented by data and control register contents, is preserved and restored over the interrupt. This data may be examined by the processor and is accessible to external devices via the master ring port providing a debug and monitor method.

Conditional Clocks

There are eight registers in SPARC for which the associated clock bits may be conditionally complemented from the control unit. These registers are the two inputs to each integer add unit, the input to file F, the input to file G, and the boolean input to each shift/boolean unit. A true condition reverses the effect of the selected clock in the same instruction. For example, if the B clock is set for the integer add unit by an instruction and the same B clock is selected for conditional execution, a true condition clears the B clock and thus prevents the loading of the B input. A false condition leaves the clock unchanged. Therefore, data may be conditionally loaded into one of these registers.

Status Bits

The control unit produces a status bit which indicates whether an interrupt is pending. A second status bit indicates a stack overflow. This second status bit must be explicitly cleared using a ring port instruction.

SPARC - PDP 11 INTERFACE

Several types of SPARC to PDP 11 interfaces were considered. These interfaces have been dedicated to communication between one host SPARC processor and the PDP 11. The design implemented for SPARC will allow interprocessor communication between the PDP 11 and multiple SPARC processors by means of a modified ring interface and the DWR-70 or DR11B general-purpose DMA PDP 11 interface.

The modified ring interface can be inserted on any standard SPARC ring communications network and transmit or receive packets to any SPARC processor on that network under program control of the PDP 11 processor. The majority of the design will be done using F100K ECL logic, allowing the design to be computer simulated using the CDC AIDS System. The interface will be located in the SPARC chassis. Data will be transmitted to the PDP 11 chassis using ECL differential line drivers. Signal level shifting will be accomplished by an interface board which will be mounted in a standard PDP 11 chassis near the DMA interface.

The ring interface will be modified to accommodate the PDP 11 data and control structure, but transfers through the interface on the ring will not be affected. Code and destination selection of the ring port will be done with eight switches; these will define one unique code out of 256 possible selections. An incoming ring packet will have its code and destination bits compared with the selected switch settings. If no compare exists, the packet will be allowed to pass through the Ring Port. If a compare is made, then the data is written into one of four 16 x 16 data FIFO's, under control of the control field, completing the packet operation. If the upper bit or interrupt bit of the control word is set when the compare operation is valid, then the control word is stored in a 64 x 4 word FIFO, which would then generate an interrupt to the PDP 11 interface.

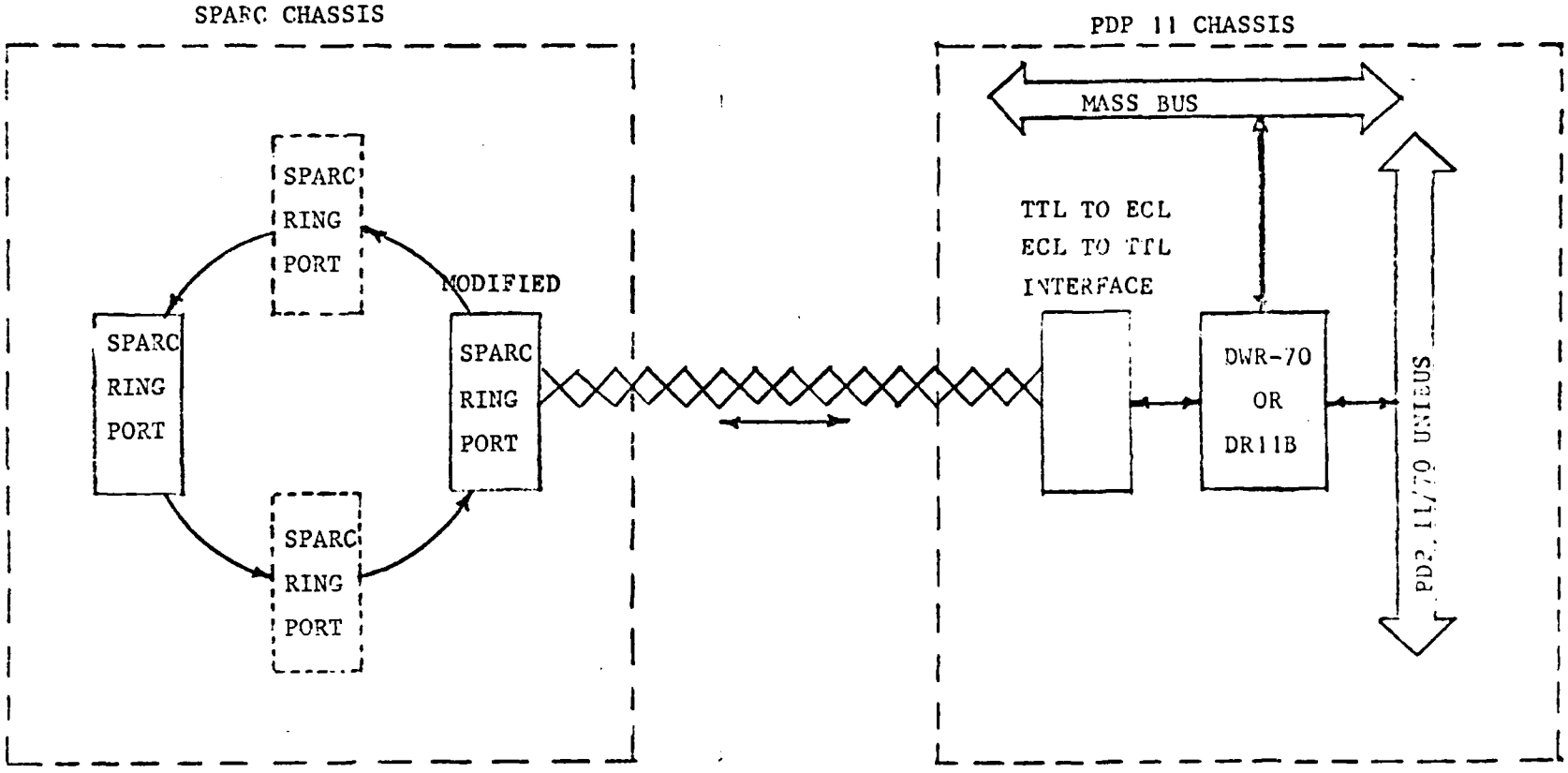

Figure 12. System Block Diagram

Z9-1081
B

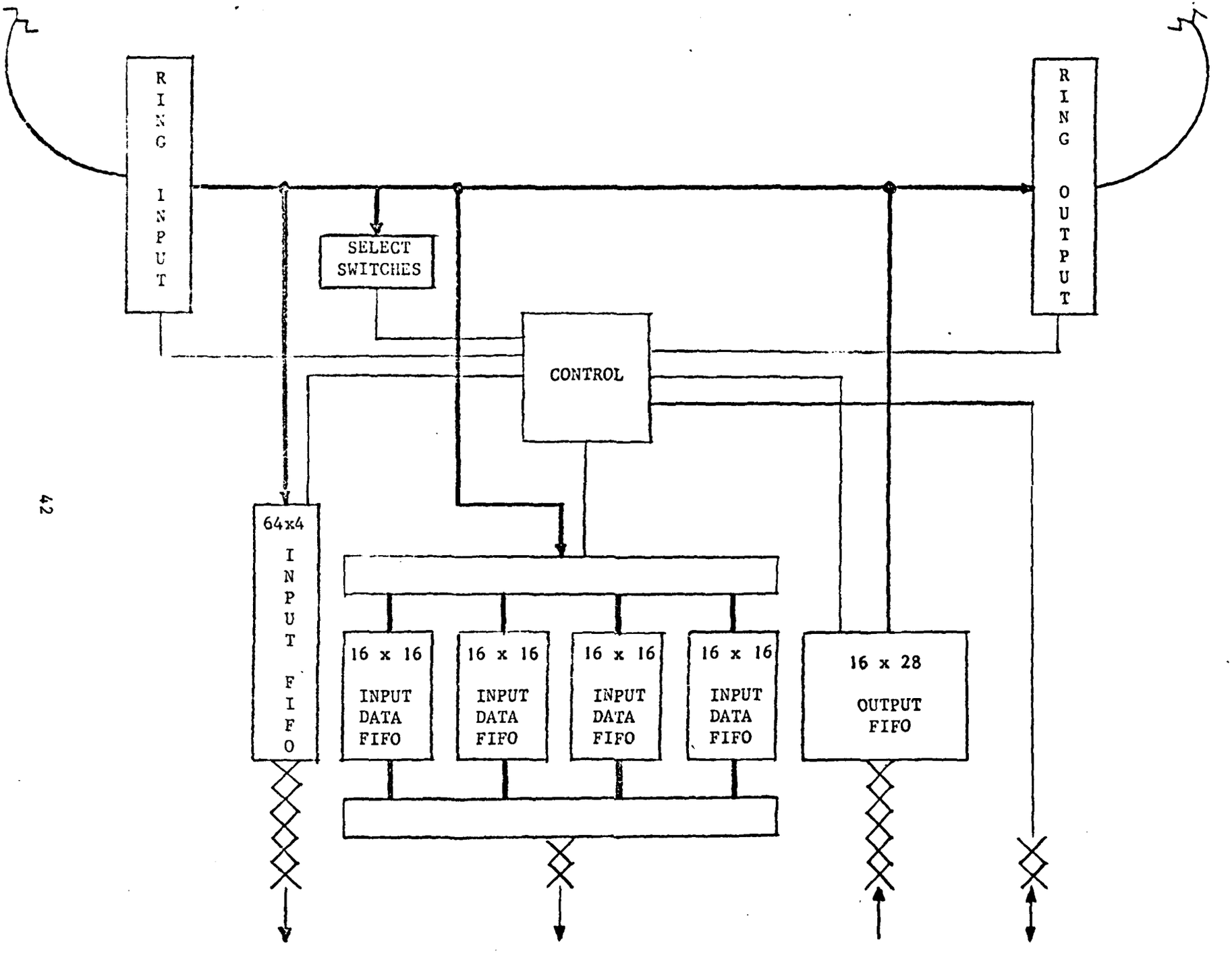

Figure 13. Modified Ring Block Diagram

Z9-1082 B

The receiving and transmitting of data is under control of the PDP 11, which must select the proper subroutine under software control to supply the word count and function control for the interface to perform the proper operations. The word count can also be selected from the first 16-bit word in the selected FIFO. The Ring packet will be structured according to the format shown in the Modified Ring Port packet structure diagram.

The three lower control field bits shall be sent to the interface status register which can be interpreted as HEX codes 1, 2, 3, or 4. These bits are used to select the appropriate FIFO and can also be used in a software routine to select separate subroutines. The FIFO's can also be selected under software control from the PDP 11.

STATUS

In July of 1979, the SPARC hardware design has been completed. All units have been simulated, and all parts are on order. Qualification samples of the new LSI arrays required have been received, and are undergoing evaluation. Approximately 25% of the other electronic components have been received. Circuit board layout is in progress, with initial types ready for fabrication. Cabinet assembly is in progress, with delivery of the initial device anticipated in August.

In the area of software, coding and debug of Version 1 of the assembler and simulator is 80% complete. Release anticipated for October. Diagnostic and early system software is on schedule for availability as required for processor debug.

Current parts delivery schedules point to the initiation of processor checkout in November.

CONCLUSION

The SPARC research project has resulted in the development of a new processing tool for image understanding work. The fabrication and operation of the two initial processors will serve as a basis, not only for software and algorithm research, but for larger processing systems consisting of arrays of SPARC processors, along with large storage facilities, and associated peripheral devices.