For these process table references the EBox supplies valid addressing information only on VMA bits 27–35. The MBox replaces VMA 13–26 with the PMA mixer 14–26 to generate a proper physical address.

2.9 DATA PATHS

The specific address and data paths in the EBox are illustrated in Figure 2-60.

The functional elements in the address path between the VMA at the MBox/EBox Interface and the primitive address source involved in forming the virtual addresses are:

Virtual Memory Address Register (VMA)
VMA Held or PC Mixer
VMA Held Register
VMA Previous Section
VMA Mixer
VMA Adder (VMA AD)
SCD TRAP Mixer
ADDER (AD)
Arithmetic Register Extension (ARXML)
Arithmetic Register (AR)
Program Counter (PC)
Microinstruction Number Field
Other Miscellaneous EBox Registers

The appropriate virtual address is formed by the VMA under explicit control of the VMA control and the microprogram.

2.9.1 Virtual Memory Address Register

The VMA is loaded during an EBox request and remains latched until the MBox responds (Figure 2-61). The VMA is a 23-bit register that accepts input from a double mixer arrangement. Thus, the incrementing or decrementing is performed in the register itself. When both VMA SEL 2 and 1 are clear, the lower mixer is enabled into VMA. The level VMA ← AD selects AD as input. The default is VMA AD as input.

In general, the VMA AD contains one of the following:

PC (18–35)
PC+1 (18–35) + (1)
PC+2 (18–35) + (2)
Process Table Address (27–35)
Fast Memory Address (32–35)

The AD contains one of the following:

Effective Address
@ Word Address
Some Special Address

The VMA Held register is loaded during each MBox memory request [MEM 02 (1)]. The left-most 12 bits of VMA Held are loaded with the request qualifiers, type of paging, context of the reference, and various other signals asserted during the request. The right-most 23 bits of VMA are preserved in VMA Held right. The contents of VMA Held are used during KL Paging mode to buffer the request state while the page fault handler sets up an MBox Page Refill cycle. This operation is generally described in Subsection 1.2.4.2, KL Style Paging and is described later in greater detail.

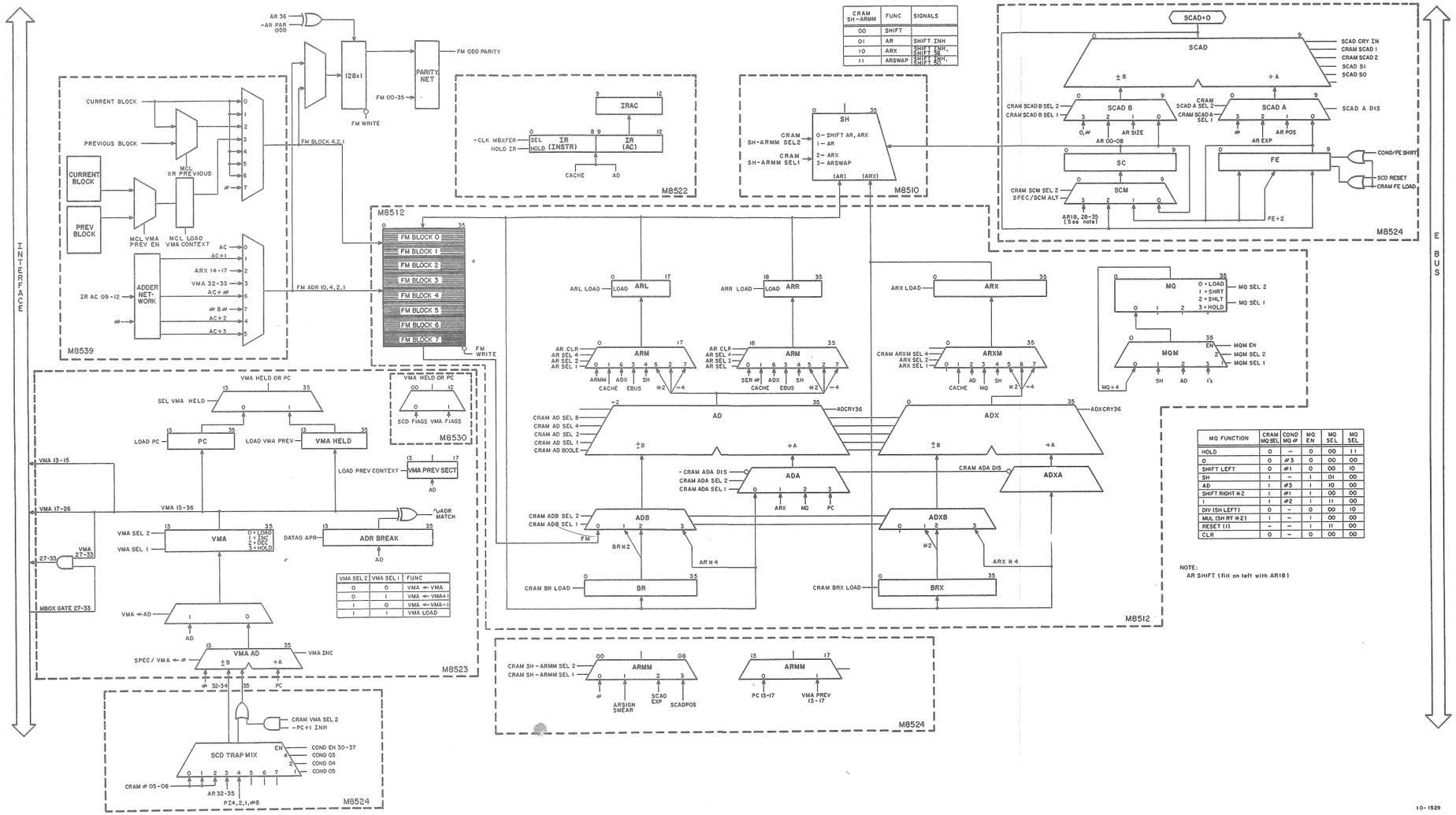


Figure 2-60 EBox Data and Address Paths

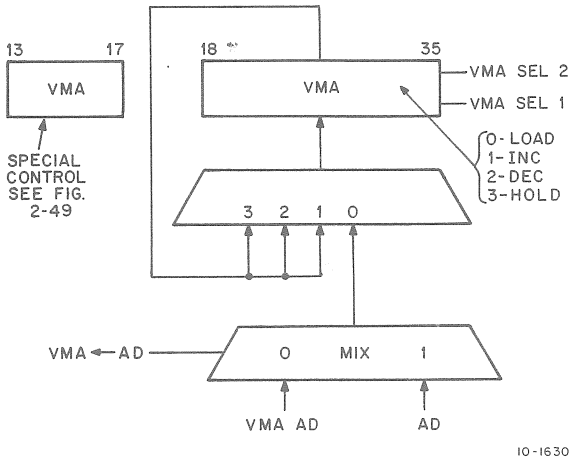


Figure 2-61 VMA Inputs

The first three selections (Subsection 3.2.1) enable the output of VMA into the VMA register for any of the following select codes:

VMA SEL 2 (0) and VMA SEL 1 (1) – Increment
VMA SEL 2 (1) and VMA SEL 1 (0) – Decrement
VMA SEL 2 (1) and VMA SEL 1 (1) – Hold

2.9.2 Program Counting

The PC is normally loaded from VMA at NICOND Dispatch, except when PI Cycle is true; this prevents alteration of PC during priority interrupt handling. When the processor is ready to fetch an instruction in sequence, the incremented PC address is supplied to VMA via the VMA AD. The VMA then supplies the address to PC. Thus, program counting is effected by the loop of PC, VMA AD, VMA, and back to the PC (Figure 2-62).

When a skip condition is satisfied, this loop is used to advance the PC during the instruction execution cycle. The PC, therefore, is automatically updated at NICOND time and if the skip is satisfied, it is updated a second time, pointing PC to the location two beyond the current location.

The PC output is available to the AD for saving a return address in a subroutine call JRST, MUUO, or similar instruction. Generally, the address saved should be for a return to the next instruction, i.e., the instruction that would have been performed had the call or jump not occurred. However, if an instruction is terminated because of a page fault or interrupt, the current address must be saved for a later return to the beginning of the interrupted instruction.

2.9.3 Loading PC

New addresses are always supplied to PC via the VMA regardless of the point of origin. The update of the PC or its inhibition is controlled by the microprogram. The following conditions cause PC+1 INH to set, inhibiting the update of PC via VMA AD:

Priority Interrupts – Setting PI Cycle
Console Instruction Execution
Halting the Processor – Halted
Performing the Trap instruction in process table location 421, 422, 423

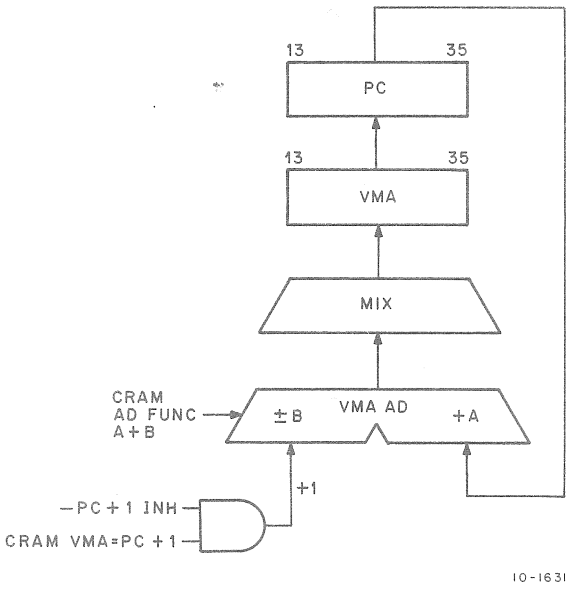


Figure 2-62 Program Count Loop

The PC is loaded at NICOND Dispatch time (Figure 2-63), providing PI CYCLE is clear. In addition, the special field function LOAD PC may also be used to load PC from VMA. During page fault handling, the SPEC/LOAD PC function is used to save the failing virtual address (VMA) in PC while saving the current PC value in ARX. Basically, the MBox builds a page fault status word in its EBus register. The physical page number is stored in bits 14–26 of this word. The EBox page fault handler must replace this address with the virtual page number in VMA 14–26 and then store the updated page fault word in user process table location 500. The operation is as follows:

Simplified Microprogram Steps Ref PF Handler

1. ARX ← old PC, PC ← failing VMA
   AR ←EBus Register; PF word

2. BRX←ARX; old PC← ARX AR; PF WORD
   AR← PC; failing VMA

3. At this time, the AR and ARX are Ref PF Handler shifted in such a way as to discard the physical page number and align the proper virtual page number in AR 14–26.

A second case is where SPEC/LOAD PC is used while halting the EBox. In this case, either a Console Halt was issued via the 10-11 interface, or a Halt instruction was performed in either user IOT mode or Kernel mode. The VMA is loaded with the current PC and the PC is loaded with the effective address currently held in VMA. At the time of the halt, the PC value in VMA points to an address one greater than the location containing the Halt and the PC contains E. PC+1 INHIBIT is set to prevent premature incrementation of the jump address now in PC.

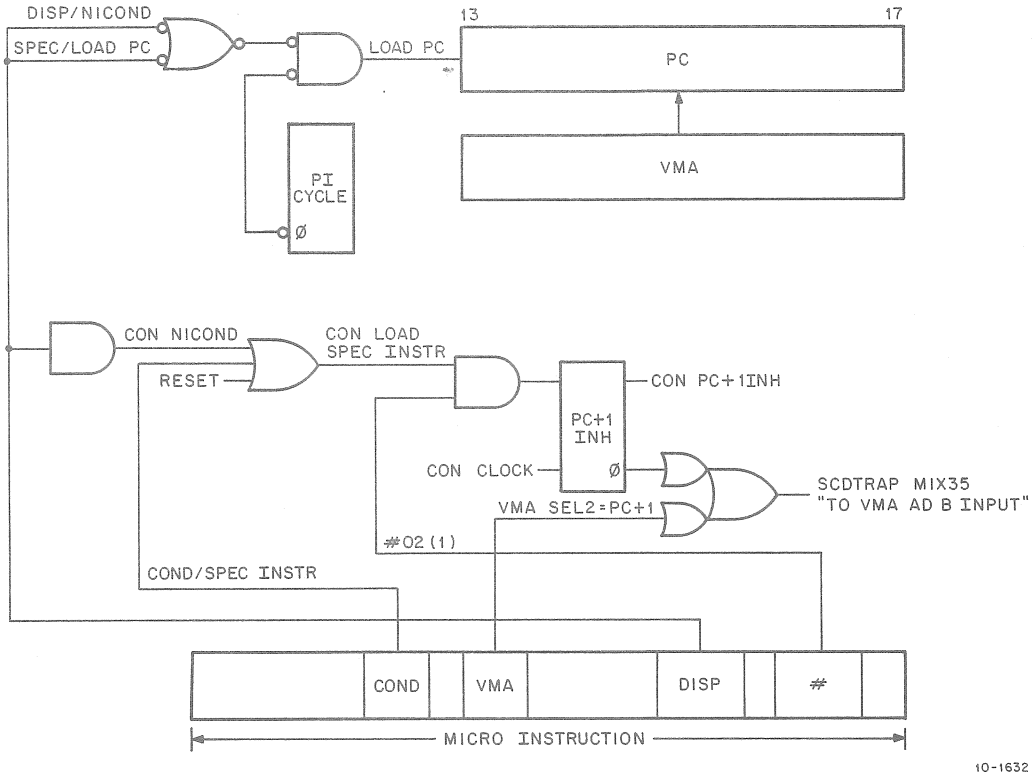


Figure 2-63 PC Loading or Inhibit

2.9.4 General Data Path Organization

The data path (Figure 2-60) is divided into four major areas, as listed in Table 2-8.

1. Fast Memory and Fast Memory Address Logic
2. Virtual Memory Address, Program Counter and related logic; 23- and 18-bit logic
3. Arithmetic logic – 36-bit logic
4. Instruction register – 12-bit logic

All of these areas derive control functions from specific fields in the microinstruction.

2.9.5 General Data Path Mixer Selection

The microinstruction or microword consists of 75 bits including parity. It is organized into variable length fields that are used to control the data path and control sections of the EBox. In the following pages each field is described functionally in terms of the particular logic with which it is associated.

2.9.5.1 AD Field – This field consists of six bits and is used to control the main adder (AD and ADX), that is constructed of type 10181 Arithmetic Logic Units. Table 2-9 lists the ALU functions. The low-order four bits specify one of 16_{10} functions. These functions are Boolean or Arithmetic as a function of bit 1 (the mode bit). If bit 1 is a one, the functions are Boolean; if zero, the functions are Arithmetic. Bit 0 is the carry in, when true it adds +1 to any Arithmetic function.

Table 2-8 Data and Address Path Breakdown

Major Area	Microfield
Fast Memory	FMADR Field COND/FM Write
Virtual Memory Addressing	VMA Field COND/VMA ← # + χ (see Note) COND/VMA DEC COND/VMA INC
VMA HELD	COND/LDVMA HELD
PC FLAGS (PC LEFT)	COND/AD Flags COND/PCF ← #
PC (RIGHT)	SPEC/LOAD PC DISP/NICOND with PI Cycle (0)
IR	COND/LOAD IR
Shift Count and Auxiliary Arithmetic 10-Bit Logic	SCAD Field SCADA Field SCADB Field SC Field FE Field
Arithmetic 36-Bit Logic and 72-Bit Logic 72-Bit Operations Require SPEC/AD Long	AD Field ADA Field ADB Field AR Field ARX Field BR Field BRX Field MQ Field SH Field ARMM Field

NOTE
χ is a constant selected by the low-order three bits of the COND code.

Table 2-9 ALU Functions

CIN	M	S_8	S_4	S_2	S_1	BOOLEAN FUNCTION	BOOLEAN CARRIES
0	1	0	0	0	0	$\overline{A}$	A
0	1	0	0	0	1	$\overline{A}V\overline{B}$	A+(A$\overline{B}$)
0	1	0	0	1	0	$\overline{A}$VB	A+(AB)
0	1	0	0	1	1	1	2*A
0	1	0	1	0	0	$\overline{A}\overline{B}$	AVB
0	1	0	1	0	1	$\overline{B}$	(A$\overline{B}$) + (AVB)
0	1	0	1	1	0	EQV	A + B
0	1	0	1	1	1	AV$\overline{B}$	A+(AVB)
0	1	1	0	0	0	$\overline{A}$B	AV$\overline{B}$
0	1	1	0	0	1	XOR	A-B-1
0	1	1	0	1	0	B	(AV$\overline{B}$) + (AB)
0	1	1	0	1	1	AVB	A + (AV$\overline{B}$)
0	1	1	1	0	0	0	-1
0	1	1	1	0	1	A$\overline{B}$	A$\overline{B}$-1
0	1	1	1	1	0	AB	AB-1
0	1	1	1	1	1	A	A-1

CIN	M	S_8	S_4	S_2	S_1	ARITHMETIC FUNCTION	ARITHMETIC CARRIES
0	0	0	0	0	0	A	A
0	0	0	0	0	1	A + (A$\overline{B}$)	A + (A$\overline{B}$)
0	0	0	0	1	0	A + (AB)	A + (AB)
0	0	0	0	1	1	2 * A	2 * A
0	0	0	1	0	0	AVB	AVB
0	0	0	1	0	1	(A$\overline{B}$) + (AVB)	(A$\overline{B}$) + (AVB)
0	0	0	1	1	0	A + B	A + B
0	0	0	1	1	1	A + (AVB)	A + (AVB)
0	0	1	0	0	0	AV$\overline{B}$	AV$\overline{B}$
0	0	1	0	0	1	A-B-1	A-B-1
0	0	1	0	1	0	(AV$\overline{B}$) + (AB)	(AV$\overline{B}$) + (AB)
0	0	1	0	1	1	A + (AV$\overline{B}$)	A + (AV$\overline{B}$)
0	0	1	1	0	0	-1	-1
0	0	1	1	0	1	A$\overline{B}$ -1	A$\overline{B}$ -1
0	0	1	1	1	0	AB -1	AB -1
0	0	1	1	1	1	A -1	A -1

NOTE: If CIN is true, add +1 to the given arithmetic function. Carry out is true if the adder, extended left, would need carry in to generate the correct function.
Carry Out is not affected by the mode (i.e., BOOLEAN FUNCTIONS give the same carry as the ARITHMETIC FUNCTIONS).

For Boolean functions, the carry in can cause a carry out if the corresponding Arithmetic function for the same S-bits would have produced a carry given the appropriate inputs. For example, assume the AD function to be performed is A and the A input equals 777777,777777. The Boolean function A performs the 1s complement of the A input, which yields a result of 000000,000000. The corresponding Arithmetic function is A and thus, if carry is true, this yields A + 1. Using the existing A input 777777,777777 +1 gives a sum of 000000,000000 and a carry. If the Boolean function A is given and carry in is true, assuming the same A input as above, the function out is 000000,000000 and a carry is generated.

The 10181 may be thought of as concurrently performing the Arithmetic operation specified and the Boolean operation specified; the sum, however, is not affected when the Boolean functions are implemented, yet the state of Carry Generate and Carry Propagate will reflect the Arithmetic result that would have formed the sum.

MC10181 Arithmetic Logic Unit Description
Figure 2-64 is an overview diagram of the ALU logic. Table 2-10 lists the ALU functions, with carry.

$$GEN = \overline{A}(S_4 B + S_8 B)$$

$$PROP = A + S_1 B + S_2 B$$

Signals GEN and PROP are used in each digit to generate the output signal Fn. In the logic mode, carries are inhibited on the output stage, and the logic function F is given by

F GEN V PROP (XOR)

(The output function is the Exclusive-Or of the two internal signals GEN and PROP).

When adding two numbers, in the absence of a CARRY IN, the Exclusive-Or function is the function required. A CARRY IN signal always complements this in this circuitry by controlling the final Exclusive-Or on the output stage.

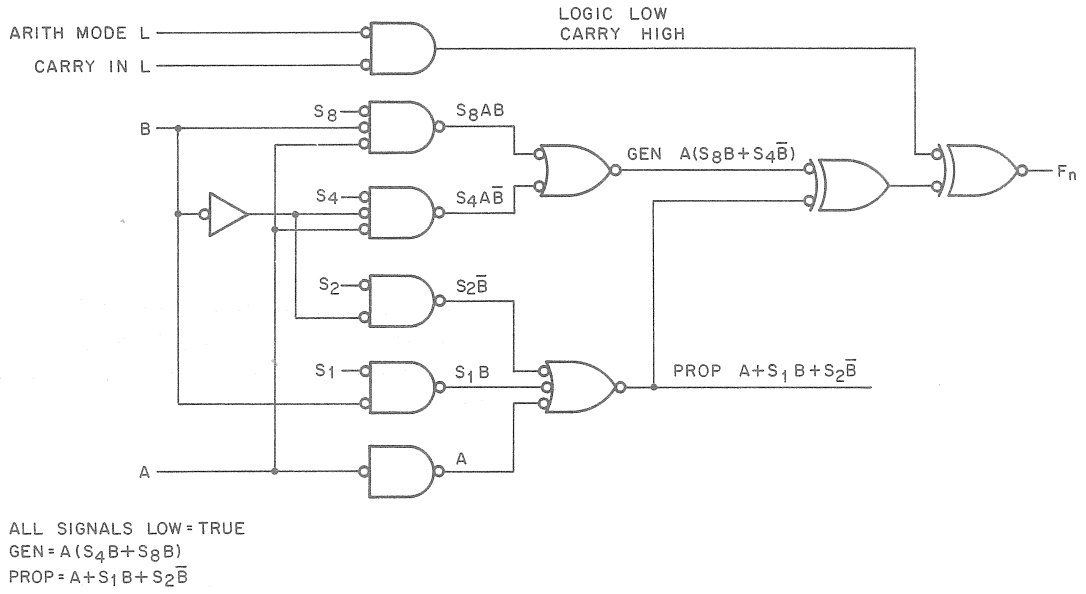


Figure 2-64 ALU Overview

Table 2-10 ALU Functions With Carry

Code				GEN	PROP	Logic Fn	Arithmetic	
S_8	S_4	S_2	S_1				CARRY LOW	CARRY HIGH
0	0	0	0	A	0	$\bar{A}$	A	A + 1
0	0	0	1	A	$A\bar{B}$	$\bar{A}\vee\bar{B}$	$A+A\bar{B}$	$A+A\bar{B}+1$
0	0	1	0	A	AB	$\bar{A}\vee B$	A+AB	A+AB+1
0	0	1	1	A	A	1	2*A	2*A+1
0	1	0	0	A∨B	0	$\bar{A}\,\bar{B}$	A∨B	A∨B+1
0	1	0	1	A∨B	$A\bar{B}$	$\bar{B}$	$A\bar{B}+(A\vee B)$	$A\bar{B}+(A\vee B)+1$
0	1	1	0	A∨B	AB	EQV	A+B	A+B+1
0	1	1	1	A∨B	A	$A\vee\bar{B}$	$A+(A\vee B)$	$A+(A\vee B)+1$
1	0	0	0	$A\vee\bar{B}$	0	$\bar{A}B$	$A\vee\bar{B}$	$A\vee\bar{B}+1$
1	0	0	1	$A\vee\bar{B}$	$A\bar{B}$	A∨B	A B 1	A B
1	0	1	0	$A\vee\bar{B}$	AB	B	$AB+(A\vee\bar{B})$	$AB+(A\vee\bar{B})+1$
1	0	1	1	$A\vee\bar{B}$	A	A∨B	$A+(A\vee\bar{B})$	$A+(A\vee\bar{B})+1$
1	1	0	0	1	0	0	1	0
1	1	0	1	1	$A\bar{B}$	$A\bar{B}$	$A\bar{B}$ 1	$A\bar{B}$
1	1	1	0	1	AB	AB	AB 1	AB
1	1	1	1	1	A	A	A 1	A

NOTE
All signals high true except GEN and PROP.

The MC10181 carries out an addition by converting the two numbers at A and B to two alternative signals GEN and PROP, given by

$$\text{GEN} = AB \qquad (S_8 = 1, S_4 = 0)$$
$$\text{PROP} = A+B \qquad (S_1 = 1, S_2 = 0)$$

For example:

	A =	0011	3	
	B =	0101	5	
then	AB =	0001	1	(GEN)
	A+B =	0111	7	(PROP)
	SUM =	1000	8	

Adding any two numbers A and B is equivalent to adding the two functions AB and A+B. However, the advantages of the second part are that one (AB) shows when carries should be generated, while the other (A+B) shows when carries should be propagated. The final sum is the XOR of the two numbers (AB and A+B), complemented by the CARRY IN signal.

$$\text{GEN} = A(S_8B + S_4B)$$
$$\text{PROP} = A + S_1 + S_2B$$

These two equations show that PROP is generated whenever A is true, which is a requirement for GEN to be true, i.e., GEN implies PROP, and thus whenever GEN is a one, PROP is also a one, and thus GEN plus PROP must generate a carry.

GEN is sufficient indication of carry generation. Similarly, PROP is sufficient indication of carry propagate.

High Logic

Actually, the circuit was designed to promote understanding for low logic, and the descriptions and tables given in the literature are far clearer for this case.

Although the circuit does give the correct answers for high logic, the circuit does operate on the low signals. Thus, an addition can be considered as an addition of the zeros, with carry generated from the addition of two zeros, and propagated, as before, by the XOR of the two numbers.

A	=	0 0 1 1 0		
B	=	0 1 0 1 0		
		1 0 0 1 1		XOR
		1 0 0 0 1		GEN
		1 1 1 0 1		PROP
COUT ←		1 0 0 0 0	←	Cin (low)
COUT ←		1 0 0 0 1	←	Carry (high)

The correct answer, therefore, occurs when Cin is asserted to the least significant bit. This can be viewed in two ways:

1. Carry is asserted high. In this case, the function considered above is Fn = A plus B and carry input adds a one. This is simple, but GEN and PROP meanings become obscure (especially when passed through the LOOK-AHEAD CARRY block).

   Generate = > (G = High and P = High)
   Propagate = > (G = High)

2. Carry is asserted low. In this case, the above function is Fn - A plus B plus 1, and the carry input subtracts a one, but hardware is simple to follow:

   Generate = > (G = Low)
   Propagate = > (P = Low)

To functionally describe the use of the various Boolean and Arithmetic functions, it is first necessary to define two other microinstruction fields which are used to enable various data to the AD A and B inputs. The first field is ADA, a 3-bit field. ADA can select the inputs shown in Figure 2-65.

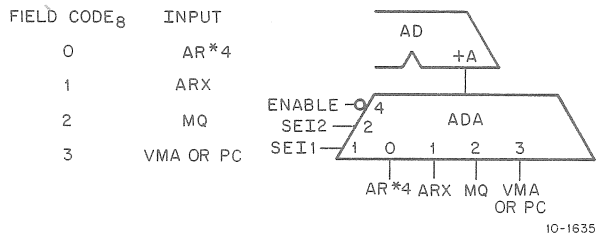
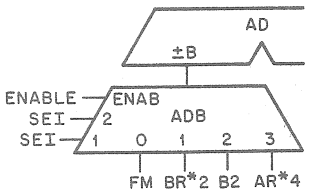


Figure 2-65 ADA Example

The second field is ADB, a 2-bit field. ADB can select the inputs shown in Figure 2-66.

FIELD CODE$_8$	INPUT
0	FAST MEMORY
1	BR X 2
2	BR
3	AR X 4

*ADR also controls ADXB, See ADB Field.



10-1634

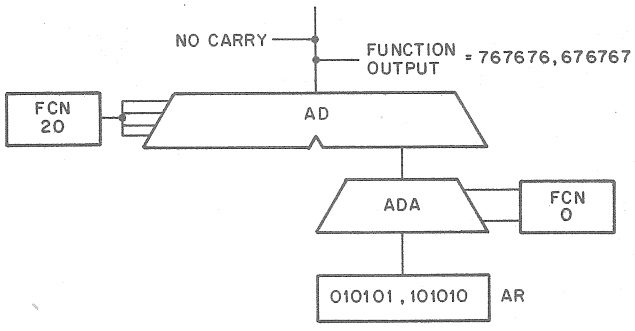
Figure 2-66 ADB Example

The following examples illustrate various operations that might be performed using EBox registers and the ADA or/and ADB input mixers. No guarantee is made that the operations illustrated are used in the microcode.

Example: $\overline{A}$ – Function 20
Initial Conditions: AR = 010101, 101010
ADA Field Function = 0

The function $\overline{A}$ performs the 1s complement of the data in AR (Figure 2-67). The AD function output is 767676,676767. Note that at this time the Carry In is false. No carries are generated in this example because the corresponding carries function is A (Table 2-9).

Example: $\overline{A}\overline{B}$ – Function 24
Initial Conditions: ARX = 777777,777777
FM = 777777,777776
ADA Field = 2
ADB Field = 0



10-1636

Figure 2-67 Function $\overline{A}$

The Boolean function $\overline{A}\,\overline{B}$ performs the logical AND of the complement of A with the complement of B (Figure 2-68). The value in ARX is selected on the ADA input mixer (777777,777777) and the value in some addressed fast memory location is selected on the ADB input mixer (777777,777776). The result presented to the function output is 000000,000000. Referring to Table 2-9, the corresponding Boolean carries function is A v B; carries are generated for the given values of A and B. For any values of A and B, no carries are generated.

Example: AB – Function 36
Initial Conditions: AR 000000,100001
BR 000765,100070
ADA Field 0
ADB Field 2

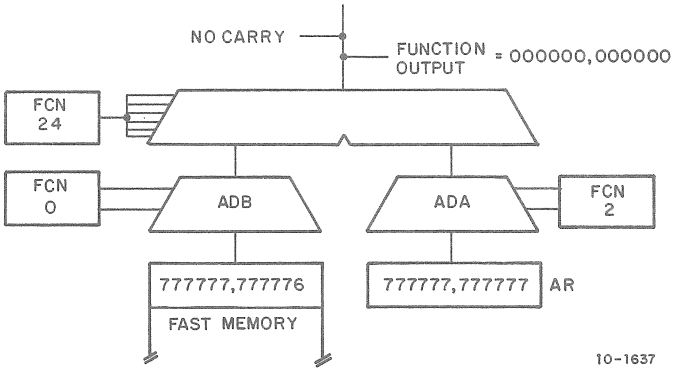


Figure 2-68 Function $\overline{AB}$

The Boolean function AB performs the logical AND of A and B (Figure 2-69). The value in AR (000000,100001) is ANDed with the value in BR (000765,100070) and the result presented to the function output is 000000,100000. Referring to Table 2-9, the corresponding carries function is AB – 1 and, given the existing inputs, it can be demonstrated that a carry from the most significant bit results if the AND of any two values results in a nonzero sum. The following demonstrates this:

```
    000000, 100001
∧   000765, 100070
    ------------------
    000000100000
+   77777 7,777777
1 ← 00000 077777
```

AB Example: A – Function 37
Initial Conditions: ARX = 000000,000100
ADA Field = 2

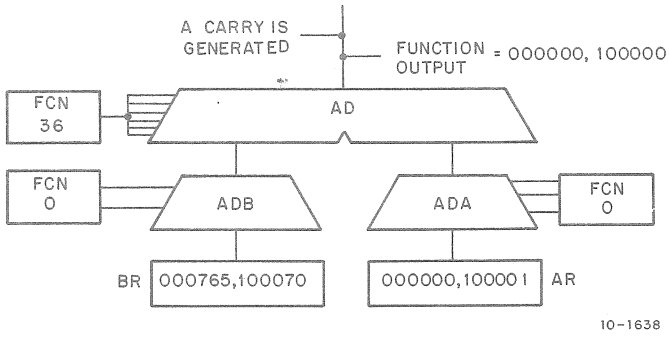


Figure 2-69 Function AB

The Boolean function A produces (at the function output) the value at the ADA input (Figure 2-70). In this example, the result is 000000,000100, but notice that the corresponding carries function is A – 1. Subtracting 1 from 000000,000100 is equivalent to adding –1, which is 777777,777777 in 2's complement notation. The result gives a carry out of the most significant bit of the AD (CRY 0). Thus, although the sum represents the ADA input 000000,000100, a carry is generated.

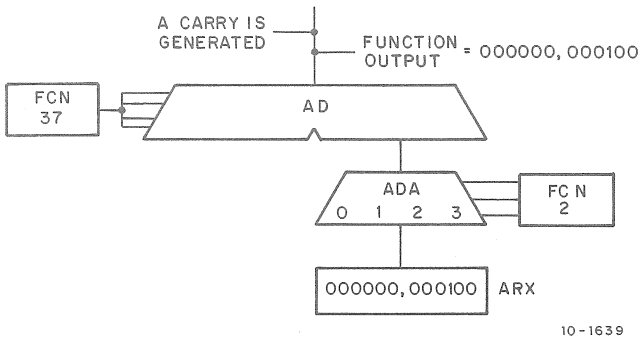


Figure 2-70 Function A

2.9.5.2 ADA Field – This field consists of three bits and is used with the main ADDER. Referring to Table 2-11, the low-order two bits select AR(0), ARX(1), MQ(2), and VMA HELD or PC(3). The high-order bit is used as a disable. This bit also controls ADXA. When the high-order bit of the ADA field is zero, ADXA selects ARX and when it is one, it selects zeros.

2.9.5.3 ADB Field – This field consists of two bits and is used in a similar fashion to that of ADA in conjunction with the main ADDER. Referring to Table 2-12, the selection is as follows: FM(0), BR*2(1), BR(2), and AR*4(3).

Table 2-11 ADA, ADXA Selection

CRAM	ADA Source	ADXA Source
0	AR	ARX
1	ARX	ARX
2	MQ	ARX
3	PC	ARX
4–7	0s	0s

Table 2-12 ADB, ADXB Selection

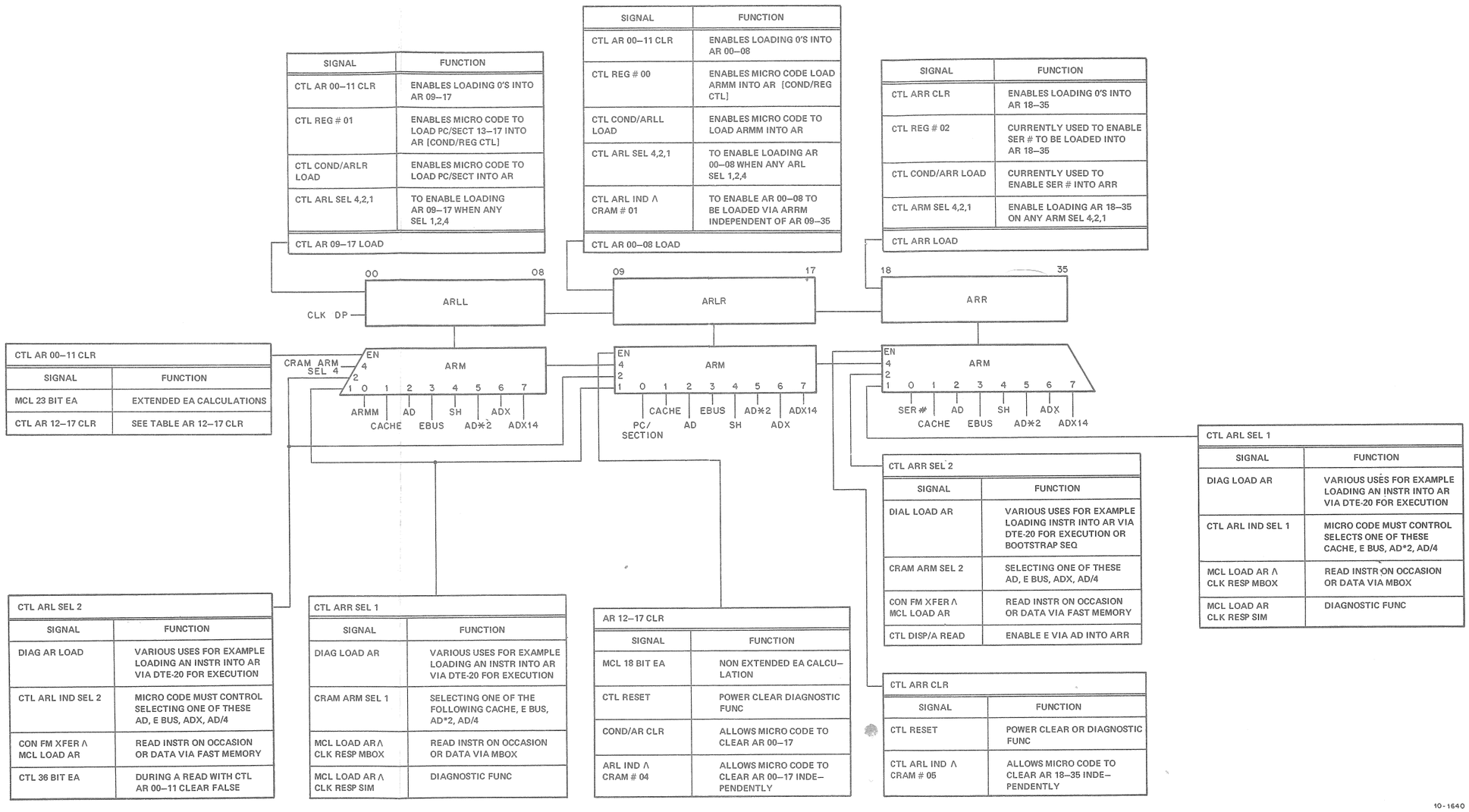
CRAM ADB	ADB Source	ADXB Source
0	FM	(unused)
1	BR*2	BRX*2
2	BR	BRX/2
3	ARX*4	ARX*4

In addition, ADB directly controls ADXB utilizing the same 2-bit field. Here the selection is unused (0), BRX*2(1), BRX/2(2) and ARX*4(3). Although AD and ADX together with ADA, ADXA, ADB, and ADXB normally function concurrently, information in ADX does not affect AD unless so specified. Carries from ADX must be specifically enabled to AD in order to affect its sum.

2.9.5.4 AR Field – This field consists of three bits. Figure 2-71 details the breakdown of various combinations of CRAM AR Selection and hardware controlled selection. Generally, the CRAM AR field specifies selection as follows: ARMM(0), CACHE(1), AD(2), EBUS(3), SH(4), ADX*2(5), ADX(6) and ADX/4(7).

AR register loading is controlled by either the hardware or microcode. Normally, the AR register recirculates its contents. Selecting any of the AR select lines CRAM ARM SEL 4, 2, or 1 enables loading AR. The selection of none of the CRAM ARM SEL lines enables the AR mixer to select ARMM. The loading of AR is then a microcode function.

During reads from core, the signal CLK RESPONSE MBOX, selects ARM SEL 1 to enable the cache data lines into AR. Similarly, on reads from fast memory via AD, FM XFER selects ARM SEL 2 to enable the AD into AR. Various combinations of clearing of AR are possible depending on the conditions. This information is given in table form on Figure 2-71.



CTL AR 00–11 CLR	
SIGNAL	FUNCTION
MCL 23 BIT EA	EXTENDED EA CALCULATIONS
CTL AR 12–17 CLR	SEE TABLE AR 12–17 CLR

SIGNAL	FUNCTION
CTL AR 00–11 CLR	ENABLES LOADING 0'S INTO AR 09–17
CTL REG # 01	ENABLES MICRO CODE TO LOAD PC/SECT 13–17 INTO AR [COND/REG CTL]
CTL COND/ARLR LOAD	ENABLES MICRO CODE TO LOAD PC/SECT INTO AR
CTL ARL SEL 4,2,1	TO ENABLE LOADING AR 09–17 WHEN ANY SEL 1,2,4
CTL AR 09–17 LOAD	

SIGNAL	FUNCTION
CTL AR 00–11 CLR	ENABLES LOADING 0'S INTO AR 00–08
CTL REG # 00	ENABLES MICRO CODE LOAD ARMM INTO AR [COND/REG CTL]
CTL COND/ARLL LOAD	ENABLES MICRO CODE TO LOAD ARMM INTO AR
CTL ARL SEL 4,2,1	TO ENABLE LOADING AR 00–08 WHEN ANY ARL SEL 1,2,4
CTL ARL IND Λ CRAM # 01	TO ENABLE AR 00–08 TO BE LOADED VIA ARRM INDEPENDENT OF AR 09–35
CTL AR 00–08 LOAD	

SIGNAL	FUNCTION
CTL ARR CLR	ENABLES LOADING 0'S INTO AR 18–35
CTL REG # 02	CURRENTLY USED TO ENABLE SER # TO BE LOADED INTO AR 18–35
CTL COND/ARR LOAD	CURRENTLY USED TO ENABLE SER # INTO ARR
CTL ARM SEL 4,2,1	ENABLE LOADING AR 18–35 ON ANY ARM SEL 4,2,1
CTL ARR LOAD	

CTL ARL SEL 2	
SIGNAL	FUNCTION
DIAG AR LOAD	VARIOUS USES FOR EXAMPLE LOADING AN INSTR INTO AR VIA DTE-20 FOR EXECUTION
CTL ARL IND SEL 2	MICRO CODE MUST CONTROL SELECTING ONE OF THESE AD, E BUS, ADX, AD/4
CON FM XFER Λ MCL LOAD AR	READ INSTR ON OCCASION OR DATA VIA FAST MEMORY
CTL 36 BIT EA	DURING A READ WITH CTL AR 00–11 CLEAR FALSE

CTL ARR SEL 1	
SIGNAL	FUNCTION
DIAG LOAD AR	VARIOUS USES FOR EXAMPLE LOADING AN INSTR INTO AR VIA DTE-20 FOR EXECUTION
CRAM ARM SEL 1	SELECTING ONE OF THE FOLLOWING CACHE, E BUS, AD*2, AD/4
MCL LOAD AR Λ CLK RESP MBOX	READ INSTR ON OCCASION OR DATA VIA MBOX
MCL LOAD AR Λ CLK RESP SIM	DIAGNOSTIC FUNC

AR 12–17 CLR	
SIGNAL	FUNCTION
MCL 18 BIT EA	NON EXTENDED EA CALCU–LATION
CTL RESET	POWER CLEAR DIAGNOSTIC FUNC
COND/AR CLR	ALLOWS MICRO CODE TO CLEAR AR 00–17
ARL IND Λ CRAM # 04	ALLOWS MICRO CODE TO CLEAR AR 00–17 INDE–PENDENTLY

CTL ARR SEL 2	
SIGNAL	FUNCTION
DIAL LOAD AR	VARIOUS USES FOR EXAMPLE LOADING INSTR INTO AR VIA DTE-20 FOR EXECUTION OR BOOTSTRAP SEQ
CRAM ARM SEL 2	SELECTING ONE OF THESE AD, E BUS, ADX, AD/4
CON FM XFER Λ MCL LOAD AR	READ INSTR ON OCCASION OR DATA VIA FAST MEMORY
CTL DISP/A READ	ENABLE E VIA AD INTO ARR

CTL ARR CLR	
SIGNAL	FUNCTION
CTL RESET	POWER CLEAR OR DIAGNOSTIC FUNC
CTL ARL IND Λ CRAM # 05	ALLOWS MICRO CODE TO CLEAR AR 18–35 INDE–PENDENTLY

CTL ARL SEL 1	
SIGNAL	FUNCTION
DIAG LOAD AR	VARIOUS USES FOR EXAMPLE LOADING AN INSTR INTO AR VIA DTE-20 FOR EXECUTION
CTL ARL IND SEL 1	MICRO CODE MUST CONTROL SELECTS ONE OF THESE CACHE, E BUS, AD*2, AD/4
MCL LOAD AR Λ CLK RESP MBOX	READ INSTR ON OCCASION OR DATA VIA MBOX
MCL LOAD AR CLK RESP SIM	DIAGNOSTIC FUNC

10-1640

Figure 2-71 AR Selection

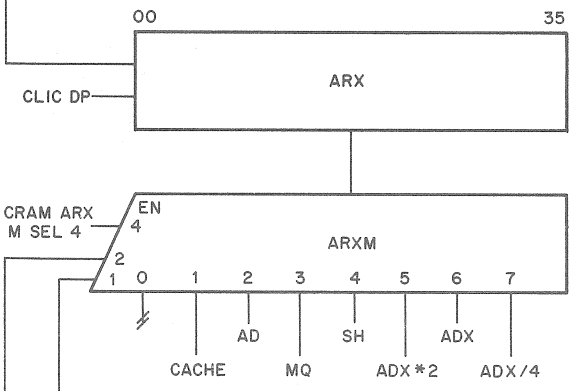
2.9.5.5 ARX Field – This field consists of three bits. Figure 2-72 details the breakdown of various combinations of CRAM ARX selection and hardware controlled selection. Generally, the CRAM ARX field specifies selection as follows: UNUSED(0), CACHE(1), AD(2), MQ(3), SH(4), AD*2(5), ADX(6), and ADX/4(7). ARX register loading is controlled by either the hardware or microcode. Normally, the ARX register recirculates its contents. Selecting any of the ARX select lines CRAM ARXM SEL 4, 2, or 1 enables loading ARX. The selection of none of these lines currently defaults to an unused input (0). As with AR, during reads from core, CLK RESPONSE MBOX, selects ARXM SEL 1, to enable the cache data lines into ARX. Similarly, on reads from fast memory via AD, FM XFER selects ARXM SEL 2 to enable the AD into ARX. Generally, the ARX is cleared via ARL IND and number 03. The various combinations are shown on Figure 2-72 in table form.

SIGNAL	FUNCTION
CRAM ARXM SEL 4	SELECTING ONE OF THESE, SH, AD*2, ADX, AD/4
CTL ARX SEL 2	SEE TABLE BELOW CTL ARXR SEL 2
CTL ARX SEL 1	SEE TABLE BELOW CTL ARXR SEL 1
CTL RESET	POWER CLEAR OR DIAG FUNC
CTL ARX CLR	SEE TABLE CTL ARX CLR

CTL ARX LOAD

CTL ARX CLR

SIGNAL	FUNCTION
CTL ARL IND ^ CRAM # 03	TO CLEAR ARX WHILE OPERA–TING ON AR



CTL ARX SEL 2

SIGNAL	FUNCTION
CRAM ARX M SEL 2	SELECTING ONE OF THESE AD, MQ, ADX, ADX/4
CONFM XFER ^ MCL LOAD ARX	READ OF INSTR, INDIRECT WORD OR DATA VIA FAST MEMORY

CTL ARX SEL 1

SIGNAL	FUNCTION
MCL LOAD ARX ^ CLK RESP MBOX	READ OF INSTR, INDIRECT WORD OR DATA VIA MBOX
MCL LOAD ARX ^	DIAGNOSTIC FUNC
CRAM ARXM SEL 1	SELECTING ONE OF THESE, CACHE, MQ, ADX*2, ADX/4

10-1641

Figure 2-72 ARX Selection

2.9.5.6 BR Field – The BR field consists of one bit and is used to select one of two possible sources as input to the Buffer Register (BR). The following sources may be selected: BR(0), AR(1).

2.9.5.7 BRX Field – The BRX field consists of one bit and is used to select one of two possible sources as input to the Buffer Register Extension (BRX). The following sources may be selected: BRX(0), ARX(1).

2.9.5.8 FMADR Field – The FMADR field consists of three bits and is used in the selection of source addresses for fast memory. Basic selection is as follows:

1. AC0(0), (IRAC 9–12),
2. AC1(1), (IRAC 9–12)+1 Modulo 16,
3. XR(2), (ARX 14–17),
4. VMA(3), VMA 32–35,
5. AC2(4), (IRAC 9–12)+2 Modulo 16,
6. AC3(5), (IRAC 9+2)+3 Modulo 16,
7. CB#(6) current ac block and selection within it is via # field,
8. #B#(7), this is some block selected by # field.

2.9.5.9 SCAD Field – The SCAD field consists of three bits and is used to control the Shift Counter Adder (SCAD) during various microinstruction operations. It is wired to implement eight functions as illustrated in Table 2-13. The input mixer structure is similar to that for the AD or ADX in that there are two input mixers labeled SCADA and SCADB. These mixers are selected via two control RAM fields labeled SCADA and SCADB.

Table 2-13 SCAD Field

CRAM SCAD			SCAD Function	Function Breakdown					
4	2	1		M	S8	S4	S2	S1	IN
0	0	0	A	0	0	0	0	0	0
0	0	1	A-B-1	0	1	0	0	1	0
0	1	0	A+B	0	0	1	1	0	0
0	1	1	A-1	0	1	1	1	1	0
1	0	0	A+1	0	0	0	0	0	1
1	0	1	A-B	0	1	0	0	1	1
1	1	0	A or B	0	0	1	0	0	0
1	1	1	A and B	0	1	1	1	0	1

2.9.5.10 SCADA Field – The SCADA field consists of three bits and is used to select various sources as input to the SCADA Input. The following sources may be selected: FE(0), AR POS(1), AR EXP(2), #(3). SCADA selections of 4–7 disable SCADA producing zeros as output.

The floating-point exponent register (FE) is a 10-bit register. The AR position field is used in byte instructions and consists of AR 00–05. The AR exponent field consists of AR bits 00–08 and the magic number field is a 9-bit control RAM field used to implement various operations. The SCADA mixer selection is shown in Table 2-14.

Table 2-14 SCADA Mixer Selection

CRAM SCADA	Source
0	FE
1	AR0–5
2	AR EXP
3	#
4–7	0s

2.9.5.11 SCADB Field – The SCADB field is a 2-bit field used to select various sources as input to the SCAD ±B input. The following sources may be selected in the SCADB mixer: SC(0), AR SIZE(1), AR00–08(2), and #(3). Selection of 4–7 disables SCADB, producing zeros as output. The SCADB mixer selection is shown in Table 2-15.

Table 2-15 SCADB Mixer Selection

CRAM SCADB	Source
0	SC
1	AR 6–11
2	AR 00–08
3	#
4–7	0s

The shift counter (SC) is a general-purpose 10-bit register used in shift counting operations such as performed in floating-point instruction and shift instruction execution. It also controls the shifter when the SH-ARMM field is zero (SH AR and ARX). The AR SIZE field is used in byte instructions and consists of AR bits 06–11. The AR00–08 is used in string and edit functions. The magic number field is a 9-bit general-purpose CRAM field used for various functions.

2.9.5.12 SC Field – The SC field consists of one bit and is used with the special field function SCM alternate. With SC and SCM alternate, four possible sources may be selected as follows:

With the special field function SCM ALT and SC field equal to zero, FE is selected. Similarly, with SCM ALT and SC field equal to one, AR SHIFT is selected. AR SHIFT consists of bits 18 and 28–35 of AR, which are derived from the effective address for shift instructions. If bit 18 is set, the shift specified is a right shift; otherwise, it is a left shift.

2.9.5.13 SH Field – The SHIFTER field consists of two bits and is used to select four possible inputs to the shifter. The selection is as follows: the combined AR, ARX(0), AR(1), ARX(2), and AR SWAPPED(3). When shifting AR, ARX left (which is the only way SH shifts physically), SC can specify up to 35_{10} shifts. Any number less than 0 or greater than 35_{10} selects ARX as output.

2.9.5.14 The AR Mixer Mixer (ARMM) – The AR Mixer Mixer (ARMM) field consists of two bits and is used with other control signals and the absence of ARM SEL 4, 2, and 1 to select various sources as input to AR mixer.

The ARMM comprises three parts: bits 00–08, bit 12, and bits 13–17. The same field that controls SH controls ARMM00–08. The following may be selected as input to ARMM00–08: #(0), AR SIGN SMEAR(1), SCAD EXP(2), and SCAD POS(3). AR SIGN SMEAR is AR0–8 from AR0. SCAD EXP is AR0–8 via SCAD, and SCAD POS is AR0–5 via SCAD.

ARMM bit 12 is controlled by CRAM SH-ARMM SEL 1 when transferring the previous section to AR for certain operations. ARMM bits 13–17 are also under control of CRAM SH-ARMM SEL 1 but the signal is actually MCL PREV SECT to ARMM. The default value for ARMM 13–17 is PC 13–17 and the selected value is VMA previous section 13–17.

2.9.5.15 VMA Field – The VMA field consists of two bits and is used to select various sources as input to VMA. The following are specified by the CRAM field VMA(0), PC(1), PC+1(2), and AD(3). Address control is presented in Subsection 2.4 and a path diagram is provided to show various combinations in Figure 2-58.

2.9.5.16 MQ Field – The MQ field consists of one bit and is used in combination with the following:

DISP/MUL
DISP/DIV
SPEC/MQ SHIFT
SPEC/REG CONTROL
MAGIC NUMBER FIELD

Refer to Figure 2-73 for various combinations.

2.10 EBOX INSTRUCTION SET FUNCTIONAL OVERVIEW

Figure 2-74 breaks down the KL10 instruction set into several functional areas. These areas are related to the minor machine cycles and to the microcode dispatch RAM decoding. The figure shows seven basic areas as follows:

1.	Group	Class of instruction
2.	Address Calculation	xr, @, B, Y
3.	Data Fetch	IMM, Read, Read-Write, Write, Read, Pse Write
4.	Execution	36-Bit Data Path (DP), 18-Bit Address Path (AP), 23-Bit AP, 10-Bit AP
5.	Special Conditions	Can cause PI, Trap
6.	Store Data	Write
7.	Interruptable	