+---------------+
! d i g i.t a l ! I N T E R O F F I C E M E M O R A N D U M
+---------------+

TO: Jupiter CPU Project Members

DATE: 4-Jan-82

CC: List

FROM: Scott G. Robinson

DEPT: LSG Hardware Engr

LOC: MR1-2/E85

DTN: 231-6988

POLE: MP18.6 (or so)

SUBJ: Jupiter FPA

The Jupiter FPA (aka APA) is an add-on to the base CPU which accelerates a subset of KC10 instructions. It is intended to make the Jupiter fit customers requiring interactive scientific computing.

1.0 ACCELERATED INSTRUCTIONS

The following KC10 instructions will be accelerated by the FPA:

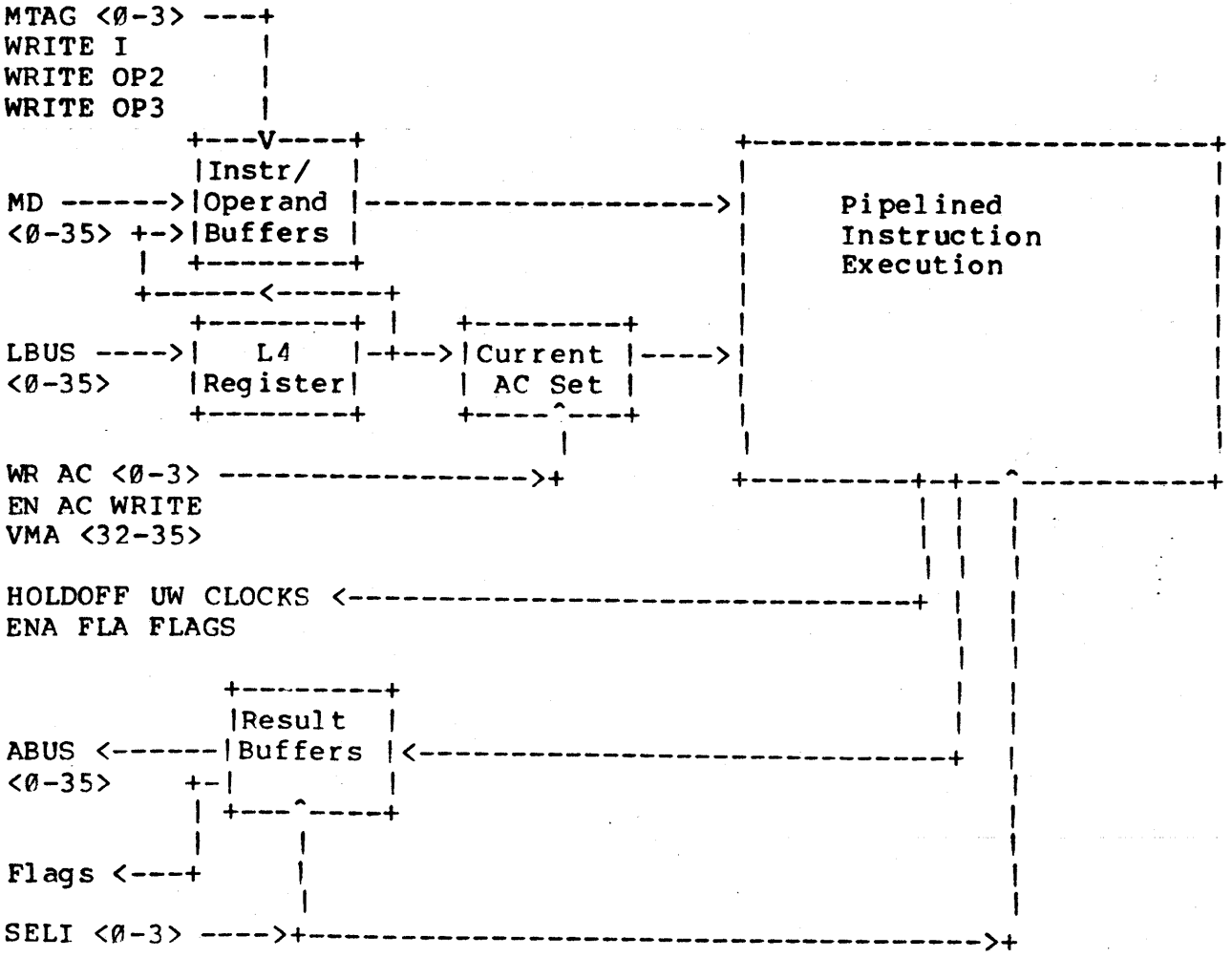
- Conversion and Scaling - FSC (132), FIX (122), FIXR (126), FLTR (127)
- Rounded Single Precision Floating - FADR? (144-147), FSBR? (154-157), FMPR? (164-167), FDVR? (174-177)
- Unrounded Single Precision Floating - FAD? (140,142,143), FSB? (150,152,153), FMP? (160,162,163), FDV? (170,173,173)
- Double Precision Floating - DFAD (110), DFSB (111), DFMP (112), DFDV (113)
- Extended Prevision Floating - EFAD (102), EFSB (103), EFMP (106), EFDV (107)
- Double Word Moves - DMOVE (120), DMOVEM (124), DMOVN (121), DMOVNM (125)
- Integer Mulitplication - MUL? (224-227), IMUL? (220-223)

All accelerated instructions will be processed directly or when XCT'd. Immediate mode instructions (FSC, FADRI, FSBRI, FMPRI, FDVRI, MULI, IMULI) will require EBOX intervention to obtain the instruction EA for the FPA because the FPA has no path to IBOX EA buffers. Instructions with indirects are treated by the EBOX as if they are XCT'd; thus, no special FPA processing is required to execute them.

2.0 EBOX/IBOX/FPA INTERFACE

2.1 Interface Diagram

The FPA Interfaces to the KC10 as shown in the following diagram:



The FPA receives Operand and AC data on MD or LBUS. The data is written into buffers maintained in the FPA. Instruction, OP2, and the AC set are redundant with other buffers in the IBOX and

EBOX. Results from FPA operation are passed on ABUS to the EBOX and written into memory or ACs when an accelerated instruction is selected by SELI. The flags are updated similar to normal EBOX ALU flags update.

2.2 Data And Flags Injection

Data and Flags Injection requires complex timing. The FPA must present the data on the ABUS for the EBOX at the correct time based upon 5 cases. The FPA must update processor flags and ensure timing for TRAP1/TRAP2 traps. In all cases below the new flags are assumed to be updated similar to an AD FLAGS or EXP TEST given during the 'First Cycle' of the EBOX. The data to be stored is passed from A-BUS to L-BUS through RF-MUX and L-MUX.

2.2.1 Single AC Store -

Single AC Stores always store at Y-AC address and can be done completely within the EBOX 'First Cycle'. The equivalent microcode is:

```
        .DCODE
; For this instruction to work RF-MUX is forced to ABUS
;  because RF-MUX is not controlled by fast ram microwords.
IC:     Y,      X1_[RF],MVE_X1,L_MVE,FPA FLAGS,LAST
```

Data is supplied by the FPA at Clock 3 so that RF can latch it at next Clock Ø and L-BUS latches RF at next Clock 1. Flags are updated at next Clock 2.

2.2.2 Double AC Store -

Double AC Stores always store at Y-AC and Y-AC +1. These stores can be done with an EBOX 'First Cycle' and an early 'Last Cycle' if X-AC is forced to Y-AC +1 during 'First Cycle' or Y-AC is incremented after 'First Cycle'.

```
        .DCODE
; For this instruction to work RF-MUX is forced to ABUS!
IC:     Y,      X1_[RF],MVE_X1,L_MVE,FPA FLAGS
        .ECODE
IC:     E,X (or Y),     RF_[ABUS],X1_[RF],MVE_X1,L_MVE,LAST
```

Data is supplied by the FPA at both Clock 3s of the above sequence. The flags are updated at ECODE Clock 2.

2.2.3 Single Memory Store -

Single Memory Stores require 'First Cycle' and an early 'Last Cycle' because of data path timing for conversion of a memory store to an AC store. The data is always stored at (EA).

```
        .DCODE
; For this instruction to work RF-MUX is forced to ABUS!
IC:     M,      X1_[RF],MVE_X1,L_MVE,FPA FLAGS
        .ECODE
; This instruction holds Memory Data and selects on WR-MUX for 22ns.
IC:     E,      X1_[RF],MVE_X1,L_MVE,WR_VMA,LAST
```

Data and flags are presented just as for the Single AC store case.

2.2.4 Double Memory Store -

Double Memory Stores require 'First Cycle' and several additional EBOX cycles because of data path timing for conversion of a memory store to an AC store. The data is always stored at (EA) and (EA+1).

```
        .DCODE
; For this instruction to work RF-MUX is forced to ABUS!
IC:     M,      X1_[RF],MVE_X1,L_MVE,FPA FLAGS
        .ECODE
; This instruction holds Memory Data and selects on WR-MUX for 22ns
;  and updates EA with assistance of the IBOX.
IC:     E,      X1_[RF],MVE_X1,L_MVE,WR-VMA,
                EA_EA+1

; This instruction stores the second memory word and holds
;  memory data and WR-MUX selects for 22 ns.
        E,M,    RF_[ABUS],X1_[RF],MVE_X1,L_MVE

        E,      LAST
```

The FPA supplies data at first Clock 3, Flags at next Clock 2, additional data at fourth (4th) Clock 3.

2.2.5 Single AC And Single Memory Store -

Single AC and Memory Stores require 'First Cycle' and several additional EBOX cycles because of data path timing for conversion of a memory store to an AC store. The data is always stored at (EA) and Y-AC.

```
        .DCODE
```

```
; For this instruction to work RF-MUX is forced to ABUS!
IC:     Y,      X1_[RF],MVE_X1,L_MVE,FPA FLAGS
        .ECODE
; This instruction stores the memory word. It also holds
;  memory data and WR-MUX selects for 22 ns.
        E,M,    X1_[RF],MVE_X1,L_MVE

        E,      LAST
```

The FPA supplies data at first Clock 3 and flags at next Clock 2.

2.3 Flags

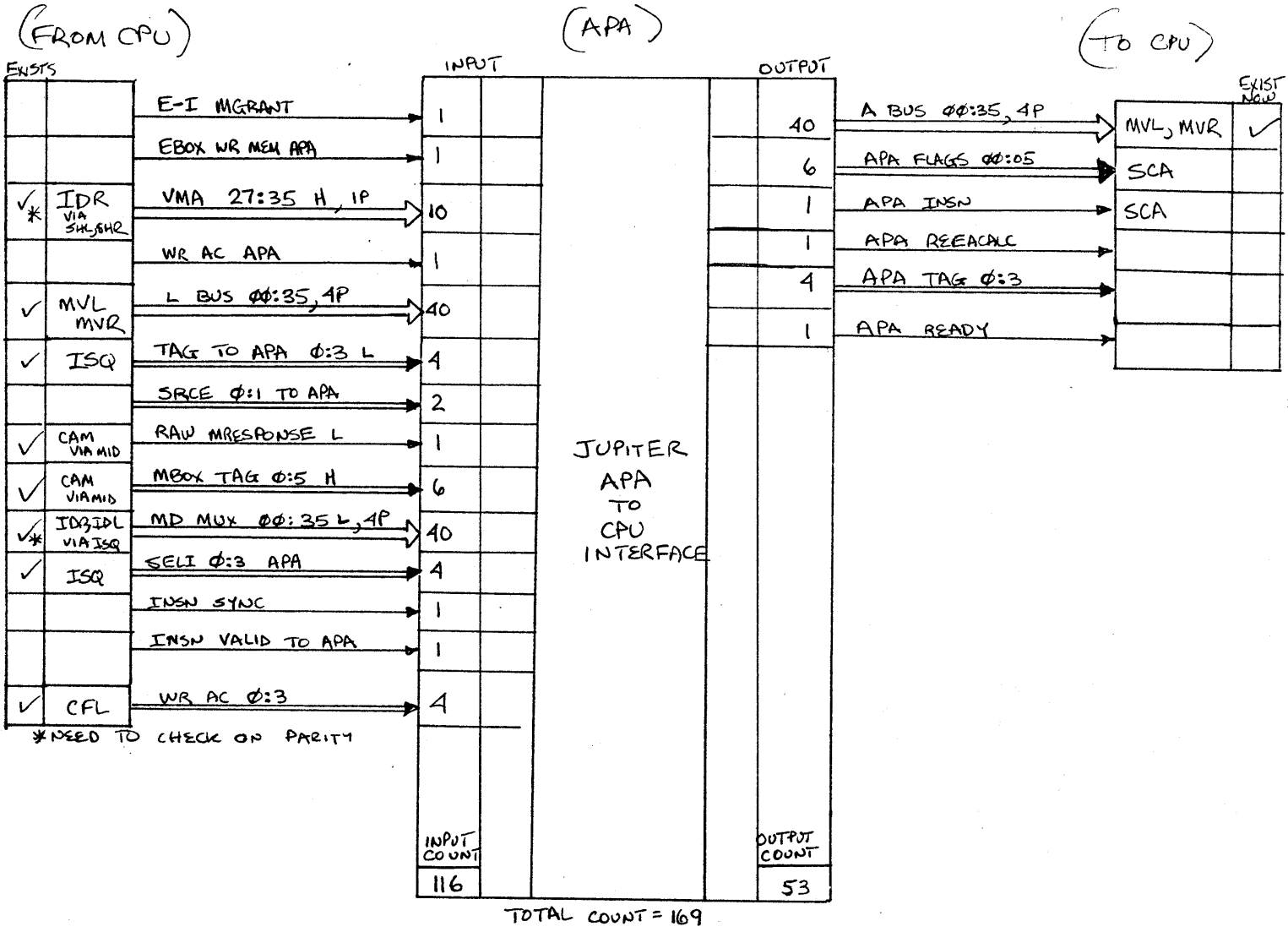
The following table lists cases of flags update required for accelerated instructions.

Case	OV	CRYØ	CRY1	FOV	T1	FUN	NODV
FP Overflow	1			1	1		
FP Underflow	1			1	1	1	
INT Overflow	1				1		
DIV Check	1			1	1		1
FP Check	1		1		1		
Zero Check		1	1				

As can be seen there are six cases and five independent bits (OV, FOV, FUN, NODV, CRY1). CRYØ is derived from (CRY1 and -OV); T1 is set directly from OV. Thus the signals for flags from FPA to SCA are:

- ENA FPA FLAGS - Set to force Flags Update during current clock 2
- OV - Overflow Flag
- FOV - Floating Overflow Flag
- FUN - Floating Underflow Flag
- NODV - No Divide Flag
- CRY1 - Carry 1 Flag

APA INTERFACE



*NEED TO CHECK ON PARITY

149 EXIST
20 NEED TO BE ADDED

Don Hooper
11/23/82
REV 2