

**DTIC FILE COPY**

**DTIC**  
**ELECTE**  
**JUL 27 1990**  
**D** *CS* **D**

(2)

Technical Progress  
OBP-80/Contract No. N00014-89-C2169

**AD-A224 618**

TO: ANDY FOX

FROM: S. ESPY

SUBJ: TECHNICAL PROGRESS FOR IDDGA - OBP UPGRADE

The following paragraphs describe progress that has been made since 5/15/90.

**TOPIC 1 - MULTIPLIER DEVICE**

The chip functionality has been defined by Martin Marietta. The following points highlight deviations from the baseline, as discussed with Alan Ross, and Ron Moody.

It was previously noted that the FA adjust control and the RND control may both be activated at the same time. When this occurs, both the Martin Marietta simulator (SIMCPU) and the Analytyx instruction set simulator (CPUSIM) produce results not consistent with the IDT device. It was determined that this mode is not useful for current OBP software. Martin Marietta recommends that the OBP Programmers Reference manual specify that the multiplier output be indeterminate when this combination of controls is applied to the device. Ron Moody suggested that, for the purposes of the OBP 80 multiplier design, we default to FA when both FA and RND occur. We are analyzing this change at present.

In last months status memo, we documented that all Control Register bit fail flags would be read out on the source bus. This was considered to be desirable since a diagnostic program could determine the functionality of the majority vote registers. We had previously baselined a decode of 0E as the Source bus decode for Control Register 3. To allow the bitfail flags to be read instead of the data, we need to provide a method of placing the device in 'test mode'. Then, when the machine instruction MOV CNTL3, R1 is issued, the bitfail flags associated with CNTL3 are connected to the source bus instead of the data.

To command the device to enter test mode, we had baselined IMR[15:13]. These bits were not defined in the current architecture. The MPY device was to have entered test mode only if all three upper bits of the IMR were set to '1'. This had the unfortunate side effect of limiting future expansion of interrupts to 13 total. Alan Ross suggested that an alternative method be found. We compromised by assigning a Test Register to Destination Bus decode 18H. Wten this register is loaded with all 1's in the top three bits, the MPY device enters test mode.

**TOPIC 2 - ALU DEVICE**

**DISTRIBUTION STATEMENT A**  
Approved for public release  
Distribution Unlimited

90 00 00 000

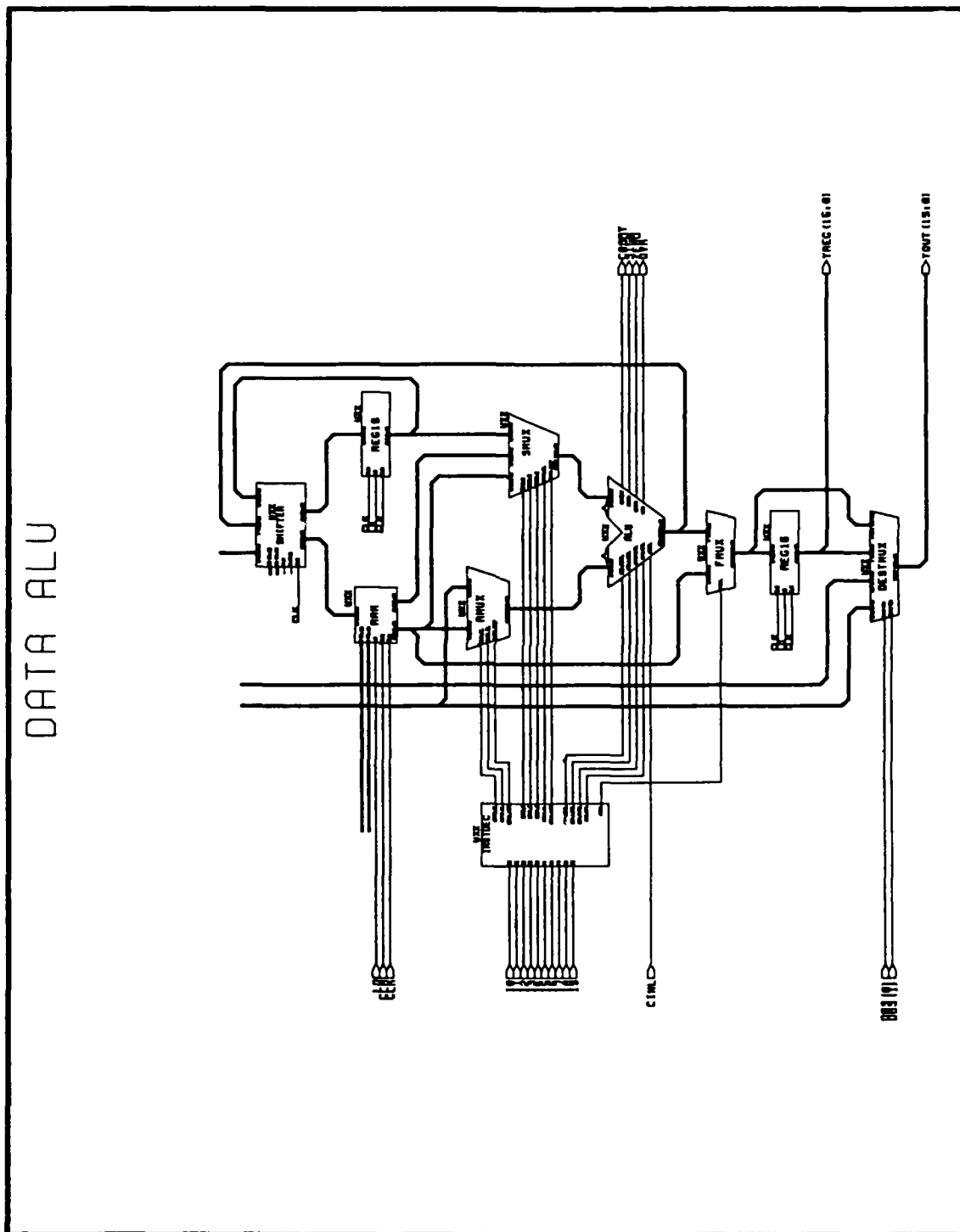


Figure 1 Single ALU Block Diagram

The run closing date for the Arithmetic Logic Unit is 9/30/90. Since this date is rapidly approaching, discussions were held regarding its design at the June TIM at Analytix. The following items should be noted:

A: The ALU is actually a twin ALU. Two 16 bit 49C402 ALU's are incorporated into a single device. These ALU's are independent of each other. Since the current OBP architecture does not support two complete ALU's, it is necessary to decode the microword. The working definition of this decode is shown in Figure 2.

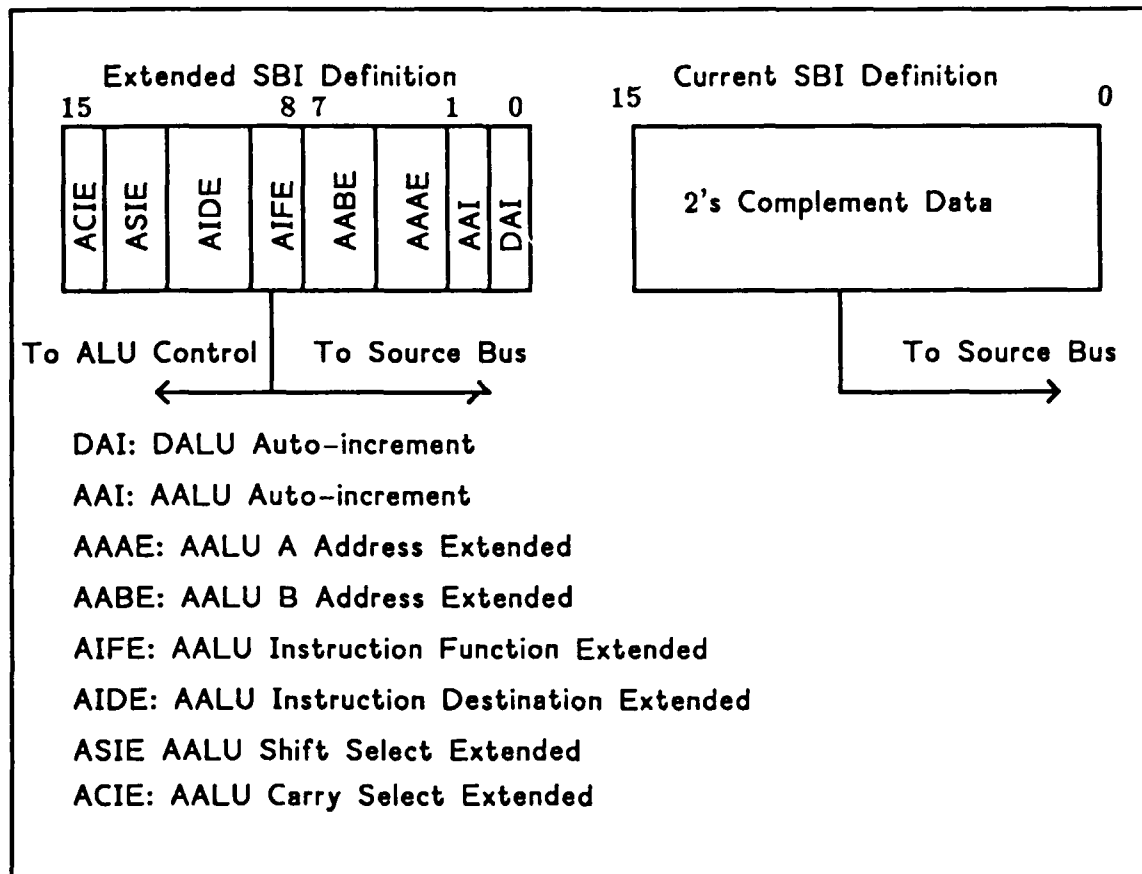


Figure 2 SBI Field Microword Extensions

The figure shows that the Source Bus Immediate (SBI) field has been re-defined to supply the added control fields for the Address ALU. These include the remaining bits of the 49C402 instruction field, the register pointer field, and the shift and carry control fields. Since it is still necessary to use the SBI field to inject 16 bits of immediate data into the processor, these extended mode definitions will only be available when a concurrency conflict does not exist.

The extended mode functions will only be available if CNTL3, bit 5 = 1. This bit is intended to be loaded at 'Boot Time'. It enables the extended decoding circuitry in the VLSI circuits. If this bit is not set, the OBP-80 will not recognize the extended definition. If it is set, the decoding circuitry will examine the Source Bus Select (SBS) field. If the current microword is

using the immediate field to inject data (SBS=6) with the SBI field, the decoding circuitry will not recognize the extended definition.

NOTE: These extended operations will require a modification to the OBP assembler to support. This is necessary since any attempt to specify the value of this field at assembly time will result in a concurrency conflict.

**B:** The ALU condition code generation has been specified by the OBP Programmers Reference Manual (Document N70425/OB87A014-03C). Section 2.3.2 of this document states: "The Carry flag and the Overflow flag are always reset by a logical operation." This is traditional, and dates back to the design of the original 2901/GPU by AMD/RCA. It is usually true that these condition codes contain no useful information for programmers.

Unfortunately, the current 'as built' OBP does not currently work as described in the Programmers Reference Manual. The 1988 version of the Integrated Device Technology Data Book (pg 8-5) specifies why this is so. For ease of physical design, the IDT designers constructed an ALU which performs logic and arithmetic operations together. This means that the carry and overflow flag generation logic always interprets the ALU results as arithmetic. To disable the flags under logic operation would have slowed the ALU critical path.

To work around this problem, Tania Fort has created an OBP flight diagnostic program which checks for condition codes as the IDT device produces them. For the OBP 80 ALU design, we have elected to go with a 'segmented ALU'. That is, the ALU is composed of separate blocks which perform arithmetic, logic, and shift operations. We chose this approach to maximize the speed of the ALU. Unfortunately, in this design the critical path slows down with the addition of logic to make the Carry and Overflow flags behave like the current system.

A block diagram of the ALU circuit macrocell is shown in Figure 3. This separation of functions is done to allow optimum performance in the arithmetic path, as it is unburdened from the task of performing other operations affecting the carry path. Therefore, when a logical operation occurs in the OBP-80 ALU design, it does not pass through the circuitry which produces the arithmetic condition codes.

The net result of this is the OBP-80 will be fully compliant with the current revision of the OBP Programmers Reference Manual where logical operations are concerned. However, we will still **FAIL** the diagnostic program which tests for condition codes as they are produced by the current OBP hardware. This will need to be changed for the OBP 80 effort.

The Martin Marietta VLSI Design Lab **STRONGLY RECOMMENDS** that the "IDT ALU specific" operations not be allowed to creep into the specification for the OBP.

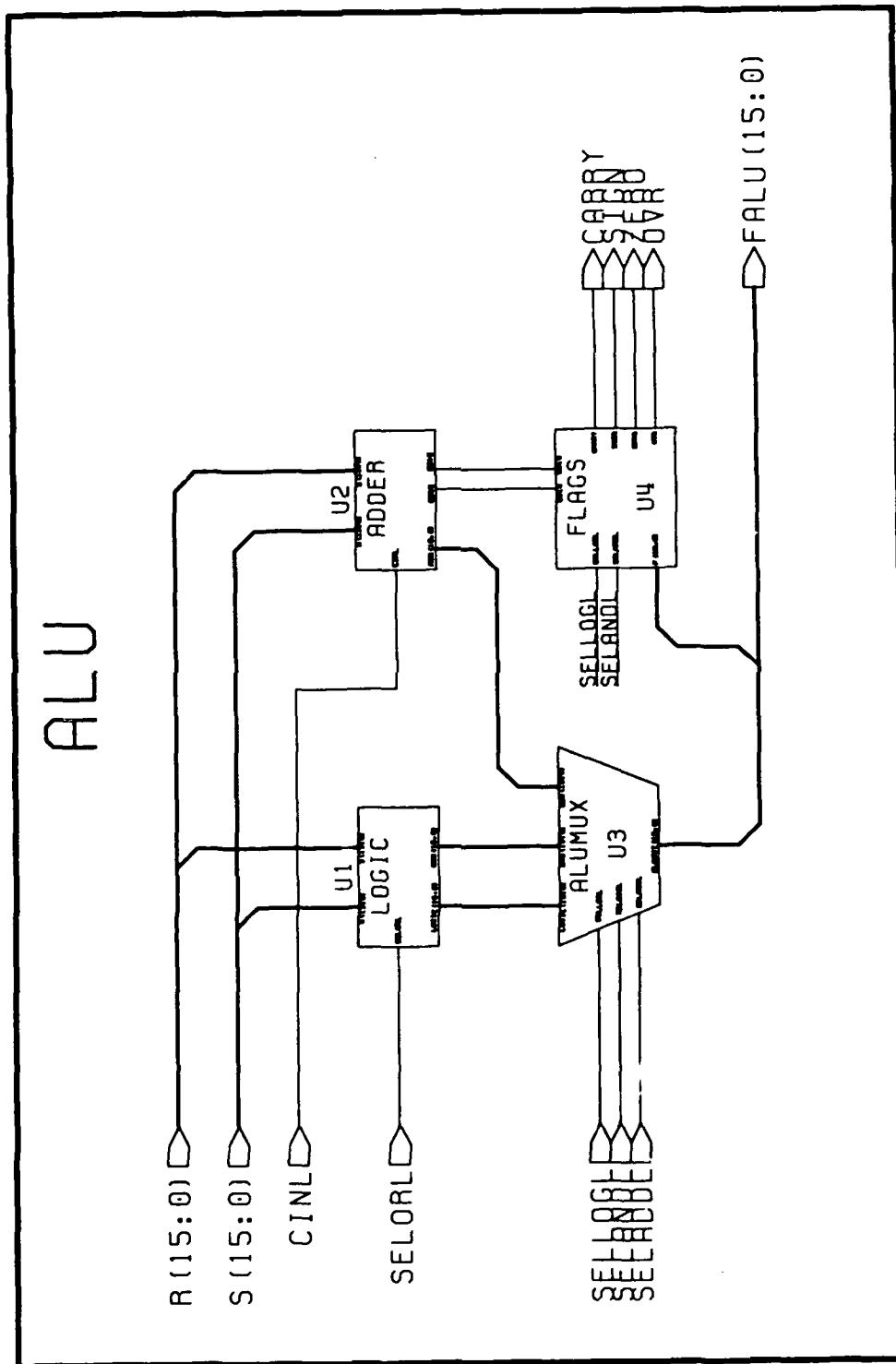


Figure 3 ALU Circuit Block Diagram

C: Several enhancements were proposed for controlling the the register portion of the ALU. In general, this is a good idea since both application execution speed and code compression can be achieved. Consider the situation where an interrupt occurs (or some other machine exception condition) and the machine state must be saved. It is desirable to save the OBP machine state as completely as possible, since the programmer has access to 100% of the machine resources. Since no registers are 'hidden' from the appication programmer, whatever resources are not saved cannot be used by the interrupt handler.

Figure 4 indicates that the only way to accomplish the saving of the machine resources is both cumbersome and inefficient. Since the microword field must explicitly specify the register to be saved, the code required to 'save the world' is enormous. To save the complete state of a single ALU requires 64 microwords.

Adding a simple register and incremter to each of the pointer fields allows the code to be reduced substantially. For instance, a loop coonstruct could be used:

```

LOOP_TOP>

WRDM0           ; write memory &

/ MOV RB+, DM0DW ; move 'B' side Reg to buffer, add 1 to pointer

/ SUBA %^X0001, AR0, AR0, DM0A ;Decrement counter

/ BRT ZERO, LOOP_TOP
  
```

```

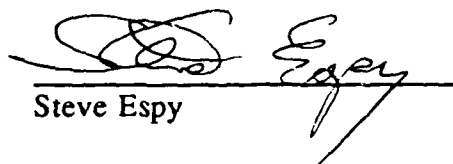
73
74                                     -----
75                                     ;Get registers ready, then transfer 1 word/cycle
76                                     -----
77 0003 002E 0000 0700 0000 0244 100E 0000  / MOV  Z^X0000,AR0           ;move first address of source
78 0004 002F 0000 0440 0000 0244 000E 0000  / MOV  AR0,DR0A             ;block to DR0 Address buffer.
79 0005 0026 0000 4000 0000 0244 000E 0000  / MOV  R0,DR0A             ;set up finished, overlap words
80
81 0006 002F 0000 4540 0001 0244 100E 0001  / WRDM0                      ;write the current word &
82 / MOV  R1,DR0DW           ;send next word to data write buffer &
83 / ADDA Z^X0001,AR0,AR0,DR0A ;increment the current address.
84
85 0007 002F 0000 4540 0002 0244 100E 0001  / WRDM0
86 / MOV  R2,DR0DW
87 / ADDA Z^X0001,AR0,AR0,DR0A
88
89 0008 002F 0000 4540 0003 0244 100E 0001  / WRDM0
90 / MOV  R3E,DR0DW
91 / ADDA Z^X0001,AR0,AR0,DR0A
92
93 0009 002F 0000 4540 000F 0244 100E 0001  / WRDM0
94 / MOV  R3F,DR0DW
95 / ADDA Z^X0001,AR0,AR0,DR0A
96
97 STOP_TAG>
98
99 000A 0026 0000 0000 0000 0244 100E 000A  / JMP  STOP_TAG
100 000B 0026 0000 0000 0000 0244 000E 0000  / NOP
101
102                                     ; Put link to OOT here.
103
104 0000 0000 0000 0000 0000 0000 0000 0000  / .END
***TOTAL ASSEMBLY ERRORS: 0
  
```

Figure 4 Screen Dump Of Current World Save

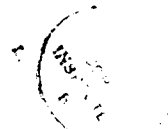
It was also suggested that a RAM bank be placed in the register field to allow mapping banks of registers. For example, the flight supervisor could initialize the RAM so that task programs would have access to 16 or perhaps 32 of the ALU registers. Achieving a context swith would be much simpler, since the interrupt handler routine could have access to the 32 registers normally hidden from the user.

Both of these ideas are under consideration, and need to be addressed at the PDR on 7/24/90.

If you have any comments or questions regarding this memo, please call me at (303) 971-9276.

  
Steve Espy

ACCOUNT FOR	
TTIS CRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By <i>per call</i>	
Distribution /	
Available to /	
Date	Approved by
<i>A-1</i>	



STATEMENT "A" Per Andrew Fox  
NRL/Code 8120  
TELECON 7/26/90

VG