ADSP-21020 Silicon Revision Information - 5/1/95

The current revision of the ADSP-21020 silicon is revision 1. The Device Identification Field (bits 31-28, the top 4 bits of the MODE2 register) can be used to determine the revision number. The Device Identification Field is 0001 for revision 1.

For further information or assistance, please call our DSP Applications Assistance Line at (617) 461-3672.

Anomalies in Revision 1

1) Interrupt Latch Bit Manipulation

If a BIT SET, CLR or TGL operation is performed on the IRPTL register (the interrupt Latch) at the same time that an interrupt arrives, the interrupt will not be latched into IRPTL. Therefore the interrupt will be missed if it cannot be serviced immediatly.

2) Latency in AC and AV status inputs to ALU and Shifter

There is a one cycle delay from the change of the AC or AV status bits in ASTAT until an instruction can use the status bits as inputs (i.e. Rn=Rx-Ry+CI; or Rn=EXP Rx(EX);). These instructions must be made indivisible by interrupts and must be delayed by one cycle minimum from any instruction that changes AC or AV. A delayed jump by +2 must placed before these instructions, for example:

```
                R0=R1+R2;               /* instruction that generates
carry */
                JUMP (PC,2) (DB);       /* makes 2 indivisible
instructions */
                R3=R4+R5+CI;            /* instruction that uses carry
*/
```

3) Hardware Interrupt Returns

If a hardware interrupt returns to a PMDA cache miss in parallel with a compute, the flag results of the compute (EQ, LE, etc.) are lost and not latched into ASTAT.

There are two workarounds that can be used:

A) If (a) one of the interrupts FIXI, FLTOI, FLTUI, or FLTII is unused and (b) "NESTM" is not enabled (_or_ within all interrupt subroutines, there no PMDA accesses in parallel with a compute or following a compute for which we care about the flags EQ, LE, etc.), then the following code can be used:

At the end of each hardware interrupt:

```
[...]
bit set stky 0x20;          /* trigger FLTII anywhere in service routine */
[...]
```

```
        rti;
        [ . . . ]

        .segment /pm fltiii; { &00000090 }          /* FLTII service routine */
      bit clr stky 0x20;     /* clear exception */
     rti;
     nop;
     nop;
        .endseg;
```

The idea behind this fix is to guarantee that a non-hardware interrupt is executed before execution is passed back to the program. If the Floating point invalid exception is being used by the program, substitute any of the other arithmetic exceptions for it. Note that the arithmetic exceptions must be triggered through the STKY register and _not_ by setting IRPTL.

C) For the general case, append the following to the end of each of the hardware interrupt subroutines:

```
        [ . . . ]

        bit clr mode1 0x1000;
        nop;
         push sts, push pcstk;   /* this ins. is available in the rev. 1.0 '020 */
         pcstk = l_end;             /* fake rti to l_end */
        rti (db);
        bit set imaskp 0x04;
        nop;

l_end:  rti (db);
        pop sts;
        nop;

        [ . . .]
```

The side effects of this is that it uses one extra location in the status stack and therefore will not work if NESTM is enabled and all five hardware interrupts could be sucessively triggered.

4) Interrupting Short-Length-Loops

If a short-length-loop is interrupted, and the interrupt routine contains a do until loop, then the short-length-loop will not operate correctly. A short-length-loop is defined by a one instruction long do until loop doing one or two iterations or a two-long-loop doing one iteration.
Workaround: Avoid the use of do until loops inside interrupt service routines.

5) Compute in RTI (DB)

On returning from a hardware interrupt (IRQx, TMZxI, VIRPT), If the last instruction in the delay field of an "RTI (DB);" instruction is an ALU or a MPY operation and the instruction being returned to is a conditional external memory access, then the conditional access may not execute correctly. Specifically, the RD or WR pins may not operate correctly.
Workaround: Do not place a compute in the last instruction of a hardware interrupt "RTI (DB);".

DOCUMENTATION CLARIFICATIONS

1) Loop aborts are now not allowed in the last three instructions of a loop.

___