# SPARC Radiation Tolerant Processor Chip Set (CCA) Design Considerations List

This document will often be released. Please refer to it regularly.

## 1. Introduction

### 1.1 Scope

This document describes the current identified specification deviations (as off October 26, 98) for the TSC691E (rev C), TSC692E (rev C) and TSC693E (rev A), with a work around proposal, when available.

### 1.2 Applicable documents

TSC691E specification, rev I, MHS September 1998

TSC692E specification, rev H, MHS December 1996

TSC693E specification, rev D, MHS April 1997

## 2. Integer Unit TSC691E

No deviation identified.

## 3. Floating-point unit TSC692E

### 3.1 FPU register addressing

#### 3.1.1 Problem description

The problem occurs in the following sequence:

Fpop1 %rs1, %rs2, %rd
up to 80 IU instructions (depending on Fpop1 and data)
lddf [], %rd
Fpop2 %rs1, %rs2, %rd

with the following conditions:
condition 1: rs2 (Fpop2) = rd (Fpop1)
condition 2: rd(Fpop1) and rd(lddf) with bit[2] = bit[4] (example f0 and f2, f8 and f10, ...)

In this case, the Fpop1 instruction will store the wrong data in the register File due to the lddf Fp instruction.

### 3.1.2 Work Around

case 1:

Source:

```
Fpop1 %rs1, %rs2, %rd
IU instructions
lddf [], %rd
Fpop2 %rs1, %rs2, %rd
```

If rd(lddf) and rs2(Fpop2) with bit[2] = bit[4] (rd[4:0], rs1[4:0], rs2[4:0])

Patch:

```
Fpop1 %rs1, %rs2, %rd
IU instructions
ldf [], %rd
ldf [], %rd+1
Fpop2 %rs1, %rs2, %rd
```

case 2: Fpop1 = fmovs or fabss or fnegs

Source:

```
movs %rs2, %rd (or fabss %rs2, %rd or fnegs %rs2, %rd)
lddf [], %rd
Fpop2 %rs1, %rs2, %rd
```

If conditions 1 and 2 are fulfilled:

Patch_1:

```
fmovs %rs2, %rd (or fabss %rs2, %rd or fnegs %rs2, %rd)
ldf [], %rd
ldf [], %rd+1
Fpop2 %rs1, %rs2, %rd
```

or Patch2 (same number of cycles):

```
fmovs %rs2, %rd (or fabss %rs2, %rd or fnegs %rs2, %rd)
nop
lddf [], %rd
Fpop2 %rs1, %rs2, %rd
```

case 3: Fpop1 is NOT equal to fmovs or fabss or fnegs or fsubs

Source:

```
Fpop1 %rs1, %rs2, %rd (with Fpop1 is NOT equal to fabss or fnegs or fmovs or fsubs)
lddf [], %rd
Fpop2 %rs1, %rs2, %rd
```

Nothing to be patched

Note: the replacement of lddf by 2 ldf works for all cases.

### 3.1.3 Example of failing code

```
!-----------------------------------------------------------------------------------------------------------
! initialization
!-----------------------------------------------------------------------------------------------------------
init:

set     data1,%l7

ld      [%l7],%fsr                      ! %fsr=0x0f080000
ldd     [%l7+8],%f8                     ! %f8=0xffffffffffffffff
ldd     [%l7+8],%f10                    ! %f10=0xffffffffffffffff
nop

!-----------------------------------------------------------------------------------------------------------
! example
! General work around:
!       replace:        ldd             [%l7+24],%f10
!       by:             ld              [%l7+24],%f10
!                       ld              [%l7+28],%f11
! Work around for this example:
!       between the following two intructions:
!                       fnegs           %f8,%f8
!                       ldd             [%l7+24],%f10
!       insert a nop:
!                       fnegs           %f8,%f8
!                       nop
!                       ldd             [%l7+24],%f10
!-----------------------------------------------------------------------------------------------------------
example:

ldd     [%l7+16],%f8                    ! %f8=0x3febab5557101f8d
nop
nop
nop

fnegs   %f8,%f8                         ! %f8=0xbfebab5557101f8d
ldd     [%l7+24],%f10                   ! %f10=0x4000000000000000 expected
fsubd   %f10,%f8,%f10                   ! %f10=0x4006ead555c407e3 expected

nop
nop
nop

std     %f8,[%l7+32]                    ! %f8 checking value
nop
nop
nop
std     %f10,[%l7+40]                   ! %f10 checking value
```

```
nop
nop
nop

end:

!---------------------------------------------------------------------------------------------------
! segment "data"
!---------------------------------------------------------------------------------------------------
.align 8

data1:
.word   0x0f080000              ! address=[%l7], %fsr init value
.word   0xffffffff
.word   0xffffffff              ! address=[%l7+8], %f8 and %f10 init value
.word   0xffffffff
.word   0x3febab55              ! address=[%l7+16], %f8 loading value
.word   0x57101f8d
.word   0x40000000              ! address=[%l7+24], %f10 loading value
.word   0x00000000
.word   0xffffffff              ! address=[%l7+32], %f8 checking value
.word   0xffffffff
.word   0xffffffff              ! address=[%l7+40], %f10 checking value (=0x0 if error)
.word   0xffffffff
```

# 4. Memory Control Unit TSC693E

## 4.1 TSC693E ERSR CPU halt indication in ERSR clearance at soft reset

### 4.1.1 Problem description

The 13:th bit (HLT) in the Error Reset and Status Register (ERSR) indicates if the IU/FPU are or have been halted. If this bit is set and a soft reset is triggered, a TSC693E internal parity error will be detected.

In case of any of the following resets:

1. Watch Dog reset

2. Software reset

3. Error reset

the reset cause is written to the ERSR and the parity is re-calculated. The TSC693E detects a parity error in this register and asserts TSC693E hardware Error. This parity error is only performed when the 13:th bit of the error and reset status register is set.

This means that if the IU/FPU were halted (by asserting the external halt signal and then resume axecution by deasserting the same signal), then the SW, WD and Error reset can’t be performed (in the future) due to parity error.

### 4.1.2 Workaround

No Workaround known.

## 4.2 UART status after UART clear

### 4.2.1 Problem description

Clearing the UARTs by setting the associated bits in the UART status register, will assign some default (reset) values to the Parity Enable, Even/Odd Parity and Stop Bits. These values are not the same values in the TSC693E Control Register and the irrespective of that register.

The UARTs enable the following when cleared:

1. Parity Enable

2. Odd parity

3. One Stop bit

It’s not possible to continue programme execution after this action, unless the incoming data has the same configuration.

### 4.2.2 Workaround

To work around the problem, re-programme the UART configuration bits in the TSC693E Control Register (bits 20:22) after each UART clear operation.

## 4.3 System Status Register update during Non-Correctable Error

### 4.3.1 Problem description

The System Fault Status Register doesn’t update the data fault type when an uncorrectable error is detected in the memory.

The memory exception handling is correct. This problem has only been observed during read operations:

1. Read byte

2. Read halfword

3. Read word

4. Read double word

The expected data fault type in the system fault register is 0x103C, but the register always shows 0x78 the reset value.

### 4.3.2 Workaround

No Workaround known.

## 4.4 Byte/halfword operations during Waitstates

### 4.4.1 Problem description

When programming the TSC693E Waitstate Configuration Register to RAM write = 0 WS and RAM read = 1,2,3 WS, Byte write operations will fail and Half word operations will fail.

This problem has been observed during IU operations:

1. stb (store byte)

2. sth (store half word)

This problem has been observed during byte/half word write in RAM on the DEM32 board and on the TSC693E VHDL model:

1. Write byte, 0 WS(write) and 1 WS(read)

2. Write byte, 0 WS(write) and 2 WS(read)

3. Write byte, 0 WS(write) and 3 WS(read)

4. Write hword, 0 WS(write) and 1 WS(read)

5. Write hword, 0 WS(write) and 2 WS(read)

6. Write hword, 0 WS(write) and 3 WS(read)

Where the TSC693E executes a read-modify-write operation.

Read 32-bit memory data, modify byte/half word, write 32-bit memory data.

The expected write strobe, MEMWR1*, is not generated by the TSC693E.

The expected write strobe, MEMWR2*, is generated correctly by the TSC693E.

Due to the missing write strobe data is not written.

EDAC check bits is written if MEMWR2* is used for check bits writing.

### 4.4.2 Workaround

No Workaround known.

## 4.5 Wrong DMA access error

### 4.5.1 Problem description

When a DMA access aborts an illegal store byte to a TSC693E register, the TSC693E register access violation leads to a DMA access error: IRL is set to 8.

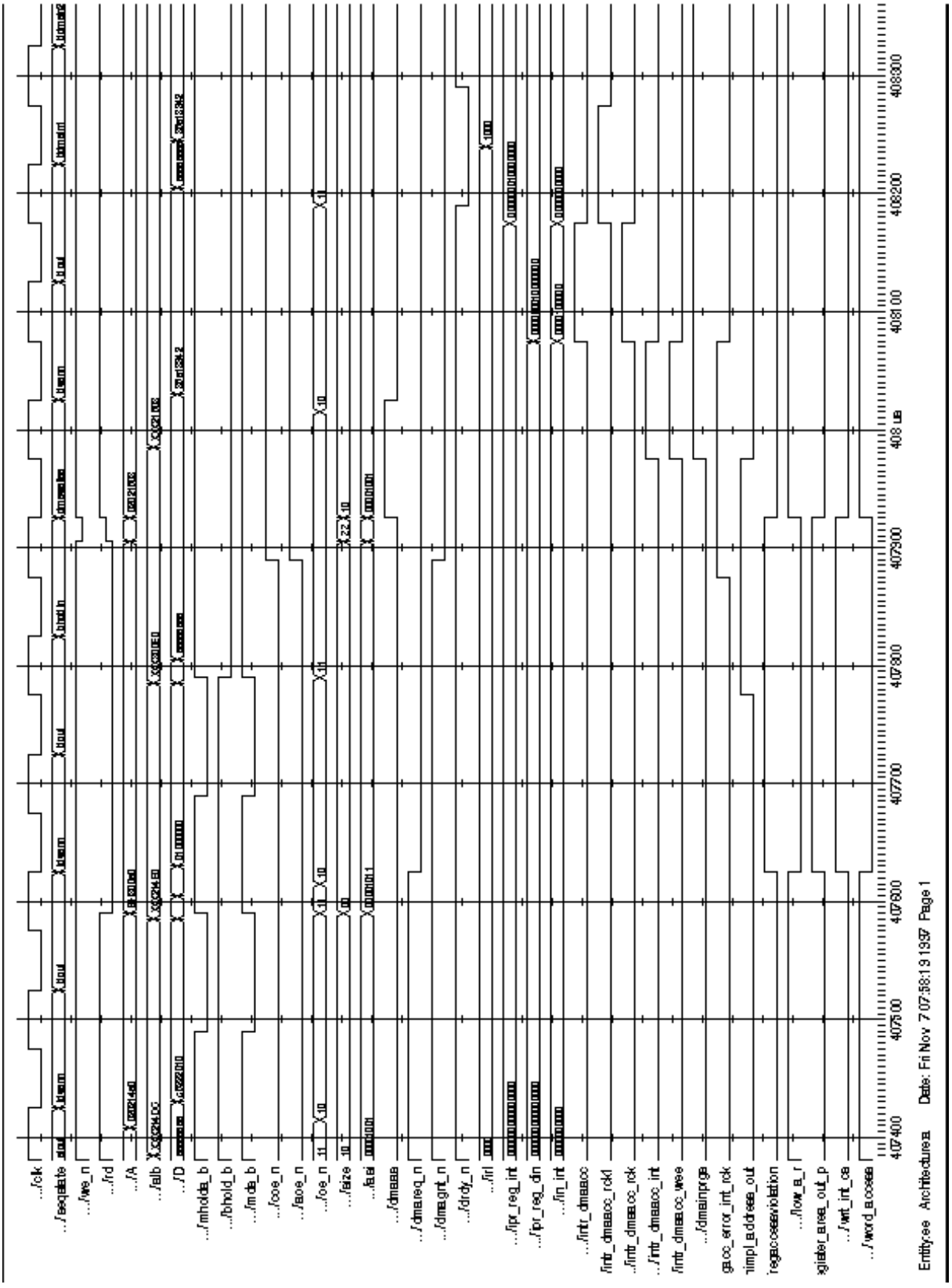
stb %g0,[%g1 + 0xe0]

with [%g1 + 0xe0] addressing TSC693E UARTA register is aborted as shown on the following timing diagram.

The TSC693E did not record the context in which the register access violation occurred, that is an IU access, and propagated the error through the DMA access context, triggering a DMA access error.

### 4.5.2 Work Around

No Workaround known.


/clk
/seqstate
/we_n
/rd
/A
/alb
/D
/mholda_b
/bhold_b
/mds_b
/coe_n
/aoe_n
/oe_n
/size
/asi
/dmaas
/dmareq_n
/dmagnt_n
/drdy_n
/irl
/ipr_req_int
/por_req_dn
/in_int
/intr_dmaacc
/intr_dmaacc_rdcl
/intr_dmaacc_rck
/intr_dmaacc_int
/intr_dmaacc_wee
/dmainprga
gaccc_error_int_rck
impl_address_out
regaccessviolation
/low_a_r
igister_area_out_p
/wrt_int_ca
/word_access
Entity:ee Architecture:a Date: Fri Nov 7 07:58:13 1997 Page 1
407400
407500
407600
407700
407800
407900
408 us
408100
408200
408300