esa
estec

european space research
and technology centre

Ref: WSD/JG/320/NL
Issue 1, 22-10-1996

ERC32 single-event upset test results

Prepared by J.Gaisler
Spacecraft Control and Data System Division
Automation and Informatics Department
ESA/ESTEC

Keplerlan 1 - Noordwijk - Netherlands
Mail address: Postbus 299 - 2200 AG - Noordwijk - The Netherlands
Tel: 071-5656565 - Telex: 39098 - Fax: 071-5656040

European Space Agency

jgais@ws.estec.esa.nl

ERC32 single-event upset test results

1 Introduction

1.1 Scope

The ERC32 radiation tolerant RISC processor is designed to be used in critical space applications where single-event upset (SEU) phenomena can not be avoided. It therefore incorporates mechanisms to detect and isolate SEU induced errors. This report consist of two main parts, the first describes ERC32 and the implemented error-detection mechanisms, while the second part provides the test results from an SEU test campaign at Brookhaven National Labs (BNL).

1.2 Acknowledgements

Many thanks to Thierry Corbier and Philippe Patron (Temic/MHS) for there effort during the tests at Brookhaven. Also thanks to John Sorensen (ESTEC) for assistance with the CREME calculations.

2 ERC32 overview

The ERC32 is a SPARC V7 compatible processing core implemented in three devices; the integer unit (IU), the floating-point unit (FPU) and the memory controller (MEC). Together they form a complete computing system to which only memory and application specific I/O needs to be added.

The ERC32 consists of three devices; a SPARC integer unit (IU), a floating point unit (FPU) and a memory controller (MEC). The ERC32 interfaces directly to external memory and IO devices. The MEC includes all system functions required to form an embedded computer and to host a real-time operating system. The most important features are:

- Address decoding
- Memory interface
- Interrupt controller
- Block protection unit
- 32-bit SEC/DED EDAC
- Two 32-bit timers
- Two UARTs
- Boot prom interface
- DMA interface
- Error manager
- Watchdog

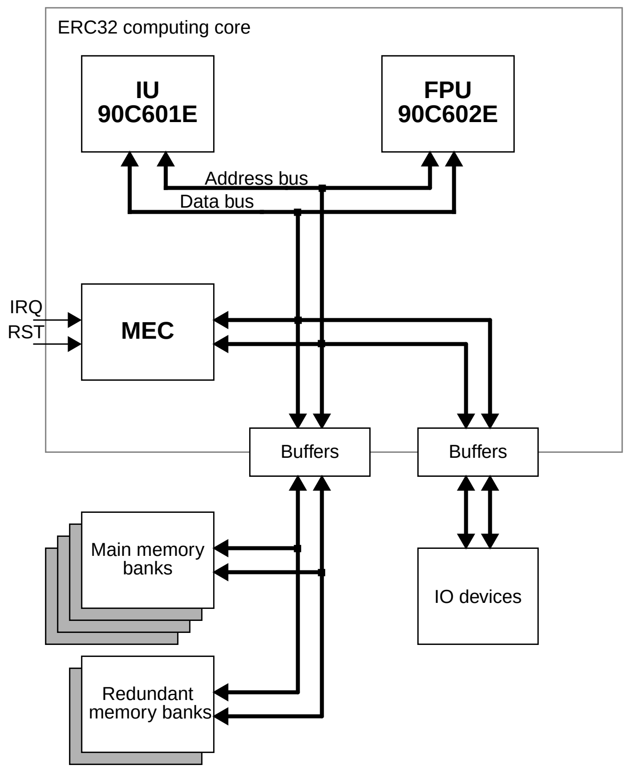

Figure 1: ERC32 architecture

The ERC32 does not use a cache memory, it runs directly from a fast SRAM-based main memory. There is no memory management unit (MMU), address translation and paging is typically not used in space-based embedded systems.

2.1 Integer unit

The integer unit (90C601E) is based on the 7C601 from Cypress Semiconductors. It has a four stage pipeline consisting of a fetch, decode, execute and write stage. A total of 140 32-bit registers are accessible to the programmer, divided into 136 general and four special purpose register. The SPARC architecture uses register windowing, the general purpose registers are divided into windows of 24 registers, with an overlap of eight. Only one window at a time is accessible, selected through the Current Window Pointer (CWP) in the Processor Status Register (PSR).

Two types of exceptions (traps) are supported, synchronous and asynchronous. The asynchronous traps are generated by external interrupts and can be masked, while the synchronous traps originate from internal events and cannot be disabled. Once a trap is taken, further traps are disabled. If a new synchronous trap occurs while traps are disabled then the proc-

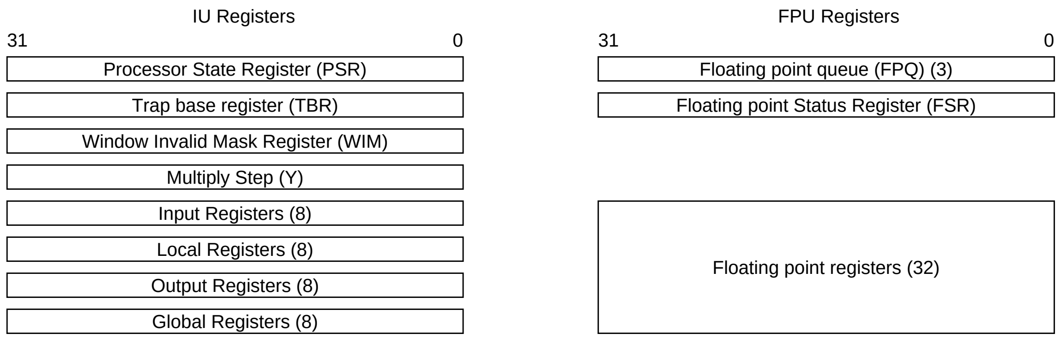
IU Registers
31
0
Processor State Register (PSR)
Trap base register (TBR)
Window Invalid Mask Register (WIM)
Multiply Step (Y)
Input Registers (8)
Local Registers (8)
Output Registers (8)
Global Registers (8)
FPU Registers
31
0
Floating point queue (FPQ) (3)
Floating point Status Register (FSR)
Floating point registers (32)

Figure 2: SPARC V7 programming model

essor enters error mode and halts. During the trap operation a new window is allocated, the program counters (PC and nPC) are copied into two local registers and the *tt* field in the trap base register (TBR) is updated to reflect the trap type. The processor then branches to the appropriate exception handling routine as indicated by the TBR.

For most synchronous traps, the trap criterion is examined during the execute stage of each instruction and the trap is taken before the instruction reaches the write stage. A trapped instruction therefore does not change the processor status. This feature is used for handling errors; when an error is detected, the failing instruction is trapped before the processor state is further corrupted.

General purpose registers. The register file, containing the general purpose registers, is provided with one parity bit per register. The parity bit is generated and written together with the data in the write stage of the pipeline. The bits of adjacent registers are physically interleaved to reduce the probability of multiple errors in one register caused by a single SEU. The three-port register file is accessed during each cycle, but the parity is only checked if the fetched values are used by the current instruction.

Special purpose registers. The four special purpose registers (WIM, TBR, Y and PSR) are divided into a number of bit fields. The bits in each bit field are updated together and each bit field is provided with one parity bit. The parity bit is generated and written when the field is updated and checked when the field is used in an operation. Some fields are used in every instruction, and are consequently checked continuously.

Temporary registers. There are several internal registers used for instruction decoding and data pipelining. A majority of those are provided with parity bits. The parity of these registers is checked during each instruction.

Table 1 shows the number of latches provided with parity bits versus the total number of latches. As can be seen, more than 98% of all latches in the IU are covered.

Module	# latches	# protected	ratio
Register file	4352	4352	100%
Main datapath	852	812	95.3%
PSR,Y,WIM,TBR	300	300	100%
Temporary registers	585	545	93.2%
Total	6,089	6,009	98.7%

Table 1: IU parity protection summary

External bus parity. To check the integrity of the external address bus an address parity bit is generated. The address bus parity is not checked by the IU since it is an output only. A parity bit on the external data bus is generated during stores and checked during loads and instruction fetches.

Three parity bits are used to protect the control buses. One bit contains the parity of the control signals going from the IU to the FPU (checked by the FPU), one bit contains the parity of the control signal going from the FPU to the IU (checked by the IU), and one bit that contains the parity of the remaining output control signals (checked by the MEC). The remaining input control signals are not protected; they can be generated by different external units, and a unified parity bit would be difficult to generate.

Program flow control. To complement the register-targeted error-detection methods, a program flow control functions is included in the IU using the embedded signature-monitor technique (ESM). The concept is to calculate a signature from each executed instruction and to compare this signature at appropriate points with a predefined checksum, to insure that the correct instructions have been executed.

Flow control is implemented by XOR-ing all instruction codes into a signature until a checkpoint instruction is reached. During the check-point instruction, the calculated signature is compared with the reference checksum (calculated by the compiler), contained in the checkpoint instruction. If a mismatch is detected, an error trap is immediately taken. The checkpoint instruction also resets the signature generator.

To preserve software compatibility with the SPARC ISA, the check-point instruction is implemented as a modified NOP. The modification is minor; the original NOP is a SETHI %g0, 0, the modified is SETHI %g0, CHK_SUM. A program using flow control is divided into branch-free blocks ended by a branch and a check-point instruction in the branch delay slot. To insure compatibility with software compiled without checksum insertion, the original NOP will disable the subsequent checking. Checking is also disabled when taking a trap or returning from a trap (RETI).

Error handling. Adhering to the general exception mechanism, the internal error-detection logic is evaluated during the execute stage of each instruction. If an error is found, an error trap is taken. The error traps are divided into six types, grouped after error location and possible recovery action (table 2).

A restartable, precise error is defined as an error which can be removed by retrying the failing instruction and for which the saved PC and nPC in the trap window indicates the correct address of the failing instruction. Recovery is performed by simply returning from the trap routine, which will resume execution at the location of the failing instruction. Errors of this type originate from parity errors in the temporary registers. When the failing instruction is retried, these registers are reloaded, and the error is effectively removed.

Non-restartable, precise errors are errors which will not be removed if the failing instruction is re-tried, but where the failing instruction is known (correctly saved PC and nPC). Removing the error will require software intervention, typically by restarting the current task. Since the failing instruction is identified, the error is isolated and will not propagate. This type of errors originate from parity errors in the user-visible special purpose registers (PSR,Y,WIM,TBR).

The most serious type of errors are non-restartable, imprecise errors. These errors are not removable by instruction retry, and cannot be tied to a particular instruction. Error isolation is still guaranteed, these errors affect all instructions and the first instruction in the trap handler will also encounter the error and will cause the IU to go to error mode and halt. A reset is the only way to recover from these errors.

Error group	Error description	Trap type
1	Restartable, precise error	0x61
2	Non-restartable precise error	0x62
3	Restartable, late error	0x63
4	Non-restartable, imprecise error	0x64
5	Register file error	0x65
6	Program flow control error	0x66

Table 2: IU error traps

Initialisation. At power-up, the register check bits are not set and have to be initialized by software. Since registers are only checked when used, no special initialisation mode needs to be entered and registers (and check bits) can be initialized in the same way as in a normal SPARC processor. Care has to be taken not to read a register before it has been written and its check bits initialised. However, using registers before they are initialized is normally not recommended even without error checking.

2.2 Floating-point unit

The floating-point co-processor (90C602E) is based on the MEIKO floating-point core. It is tightly coupled to the integer unit which fetches and decodes all floating-point instructions. The floating-point instructions are started by the IU using the INS1/INS2 signals and then execute independently inside the FPU. Most FP instructions execute in parallel with IU operation. During the parallel execution, the address and data of the current instruction is held in the floating-point queue, FPQ. If an exception (e.g. overflow) occurs during the execution of an floating-point instruction, the FPU will assert $\overline{\text{FEXC}}$ and enter pending_exception state. The IU will recognize the floating-point exception at the start of the following floating-point instruction and take a floating point trap. The exception type will be indicated in the *ftt* field in the floating-point status register.

The FPU error-detection scheme is similar to the IU scheme. All registers are provided with parity bits, and FPU generates and checks parity bits for all buses (address, data and control). The error-handling is slightly different; the FPU cannot handle the detected errors on its own, but flags them to the IU which have to take corrective measures. When and error is detected, the FPU enters pending_exception state and indicates the error type in the *ftt* in the floating point status register. Three types of errors are defined; restartable error, non-restartable error and data bus error.

A restartable FPU error is defined as an error which was detected before the FPU state was changed, and where the failing instruction can be re-executed. These error originate from instruction decode and data pipeline registers, or from a control bus parity error. A non-restartable error is an error which was detected after the FPU state was changed, and can therefore not be removed be re-executing the instruction. A data bus error indicates a bus parity error during a floating point load instruction. This error does not affect the FPU state, but since FP loads do not enter the floating-point queue the instruction cannot be re-executed.

Module	# latches	# protected	ratio
Register file	1024	1024	100%
Datapath	756	756	100%
Temporary registers	406	406	100%
Total	2,186	2,186	100%

Table 3: FPU parity protection summary

At power-up, the register check bits are not set, and have to be initialized by software. As for the IU, no special initialisation mode needs to be entered and the registers (and check bits) can be initialized in the same way as in a normal SPARC FPU. Again, care has to be taken not to read a register before it has been written and its check bits initialised.

2.3 Memory controller (MEC)

The MEC implements important system support functions such as chip select decoding, wait-state generation, EDAC, timers and USARTs. Error-detection is implemented by providing each MEC register with a parity bit which is continuously checked. The parity of the external address, data and part of the control bus is checked by the MEC.

The MEC also contains an error manager, where the error signals from the IU, FPU and the MEC itself are sensed. For each error type, the error manager can be programmed to either ignore the error, issue an interrupt, reset the ERC32 or halt.

2.4 Error-detection overhead

The introduced error-detection scheme has a relatively low overhead; less than 15% in terms of silicon area. The timing impact is basically the maximum delay through a 32-bit parity generator, approximately 8 ns on a 1 µm CMOS technology. The flow control scheme gives a negligible hardware overhead, but results in a run-time performance degradation. If a program consists of 10% branches, and the schedulability of the delay slot is 50%, the total performance degradation is 5% (0.1 * 0.5). Five additional pins are added to the IU and FPU for bus parity and error flagging.

3 ERC32 SEU test campaign

3.1 Objectives

Apart from the traditional objectives of determining the LET threshold and cross-section for the three devices, a major objective was to asses the efficiency of the on-chip error-detection mechanisms. This called for a test system where the ERC32 would operate under typical conditions but where the handling of the induced errors could be monitored with high accuracy. The following parameters were to be determined:

- LET threshold and cross-section
- Coverage of on-chip error-detection mechanisms in IU, FPU and MEC
- Coverage of system level error-detection mechanisms

3.2 Test system

The ERC32 SEU test system consist of a host computer and a target system connected together via a serial RS-232 link. The host computer consists of an ordinary PC with monitoring software that logs events in the target system. The target system consist of an ERC32 chipset, 2 MByte RAM and 512 Kbyte flash-PROM. The IU and FPU are running in master/checker mode, i.e. two devices in parallel. The target system is configured to use all error-detection functions available; parity on address-, data and control buses is checked and the memory is provided with EDAC checkbits. The target system uses UART A in the MEC to connect to the host system at 19200 baud.

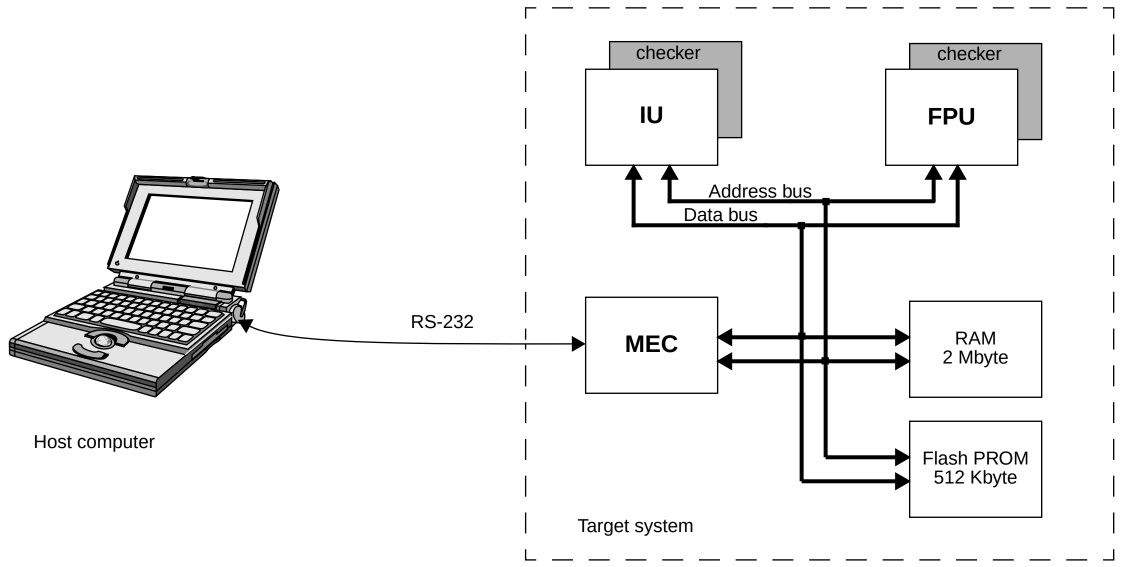

Figure 3: Test system architecture

3.3 Test methodology

During SEU testing, one device at a time is radiated. For the IU and FPU, the master device is radiated and the checker is used as a reference to insure detection of all error events. SEU induced errors in the IU and FPU are detected in three ways:

- hardware error trap in master IU
- hardware FP exception in master FPU leading to an FPU exception in the IU
- comparison error flagged by checker (IU or FPU)

The MEC cannot work in master/checker mode and errors are detected through a combination of on-chip error-detection mechanisms and software techniques (self-checking programs).

The target system was running at 10 MHz. No special latch-up protection circuit was used since earlier tests performed by CNES did not show any latch-up. Only one device of each type was radiated. The devices had no special markings and were manufactured on the standard SCMOS 0.8 RT process. The ambient temperature during the tests was approximately $20^{o}C$, the supply voltage +5.0V.

3.4 Target system software

The target software consists of a loader and three applications; iureg, fpureg and paranoia. After power-on reset, the loader performs the following:

- clear all IU and FPU registers to initiate parity bits
- clear all memory (RAM) to initiate EDAC checkbits
- initiate various MEC registers
- load all three applications in RAM

The loader then waits for input from the console (UART A) to start one of the applications. For down-loading and debugging purpose, sparcmon can also be started.

The *iureg* program scans the IU register file to detect SEU induced errors. I works as follows: most registers are set to 0x55555555 and then XOR-ed together. If the result is not equal to 0x55555555, the program writes #FIU (Failed IU test) on the console. After each 25,000 successful iterations, #PIU (Pass IU test) is written on the console. Before the start of the test, all register windows are flushed so the no window overflow/underflow traps occur during the run. Detected errors should therefore not occur when traps are disabled and be properly reported. Only 123 registers (of 135) in the register file are tested since some register are used by the compiler an must not be modified, and the 8 local registers in the invalid window cannot be reached.

The *fpureg* program checks for errors in the FPU register file and for errors in FP operations. FADDD and FSUBD is performed between all registers and a checksum is calculated. If the checksum is correct, #PFU is reported, otherwise #FFU.

The famous *paranoia* floating-point validation program is used as a combined integer/floating-point application. The program performs numerous test to validate the floating-point handling of a processor. All FPU calculations are verified against values calculated in the IU. The paranoia program consists mostly of integer instructions; only 5% of the executed instructions are floating-point. This makes the program suitable as a combined IU/FPU application. In addition, the program is almost totally self-checking; any undetected errors in the FPU calculations would be detected as a FPU failure by the software cross-checking. Undetected errors in the IU would not always be detected. However, due to the continuous cross-checking between IU and FPU results, it is believed that most undetected IU errors would result in an IU/FPU cross-check error. Paranoia executes one iteration in about 150 ms. After four correct iterations, #PPA is written to the console. If a error is detected by the program, #FPA is written.

All three applications are running on top of a small run-time kernel that in addition to the normal window handling routines also provide support for error detection and reporting. Any unexpected trap will be reported to the host system via the console. After the trap has been reported, a software reset is issued to re-synchronize the master and checker devices, and the application is restarted. The application is not reloaded from PROM into RAM in this case to minimize the time required for restart. However, the IU and FPU register files are cleared to remove any parity errors. The MEC is always cleared by the reset. The time to report the error and restart the application is less than 10 ms at 10 MHz operation.

3.5 Error reporting and classification

The following reports are provided by the target system run-time kernel:

Report code	Report description
#RP	Power-on reset
#RS	Software reset
#RE	MEC error manager reset
#RW	Watchdog reset
#M	Loader started successfully
#T**nn**	Trap <**nn**> (hexadecimal) occurred. All traps except window handling routines are reported. An FP exception trap is reported as 'TF**n**', where <**n**> indicates the ftt field in the FPU status register.
#Inn	Interrupt 1 occurred. This interrupt is generated if a masked error is detected by the MEC error manager. This will typically only happen if an master/checker compare error occurs without causing a error trap in the IU. The <nn> field indicates the LSB part of the MEC error status register.
#Dnn	A data or instruction fetch trap occurred triggered by an external MEXC. Bit 7..4 of <nn> indicates the MEXC the data fault type in the MEC system fault status register, bit 3 is IU data fault valid and bit 0 is asynchronous fault valid.
#FIU	IU test program failed
#FFU	FPU test program failed
#FPA	Paranoia test program failed

Table 4: Target system error reports

The SEU induced errors are classified in two types: detected and undetected. Undetected errors are errors that were not detected by the hardware error-detection mechanisms and only detected by the checker devices (IU or FPU) or by software failure reports. These error corresponds to the error reports #FIU, #FFU, #FPA and #Inn.

Interrupt 1 is used to report undetected hardware errors. This interrupt is generated by the MEC when a hardware error is flagged by a master device or a master/checker error is flagged by a checker device. If the error occurred in the master IU and was detected, an error trap is taken and the handler for interrupt 1 is never reached since the error trap has higher priority. If the handler for interrupt 1 *is* entered, an undetected error (master/checker) must have occurred since no error trap was taken by the IU. Errors for the FPU are treated slightly different; due to the parallel execution of the FPU, an detected error in the FPU does not lead to an IU trap until the next FPU instruction. If interrupt 1 handler is entered and the only error in the MEC error status register is FPU HW error, the interrupt is cleared and execution of the program is continued. The IU will take a FP exception trap on the next FP instruction and report the error.

4 Results

4.1 LET and cross-section

The following LET values and cross-sections were obtained during a test campaign at Brookhaven National Labs (BNL) on 30-31 August 1996.

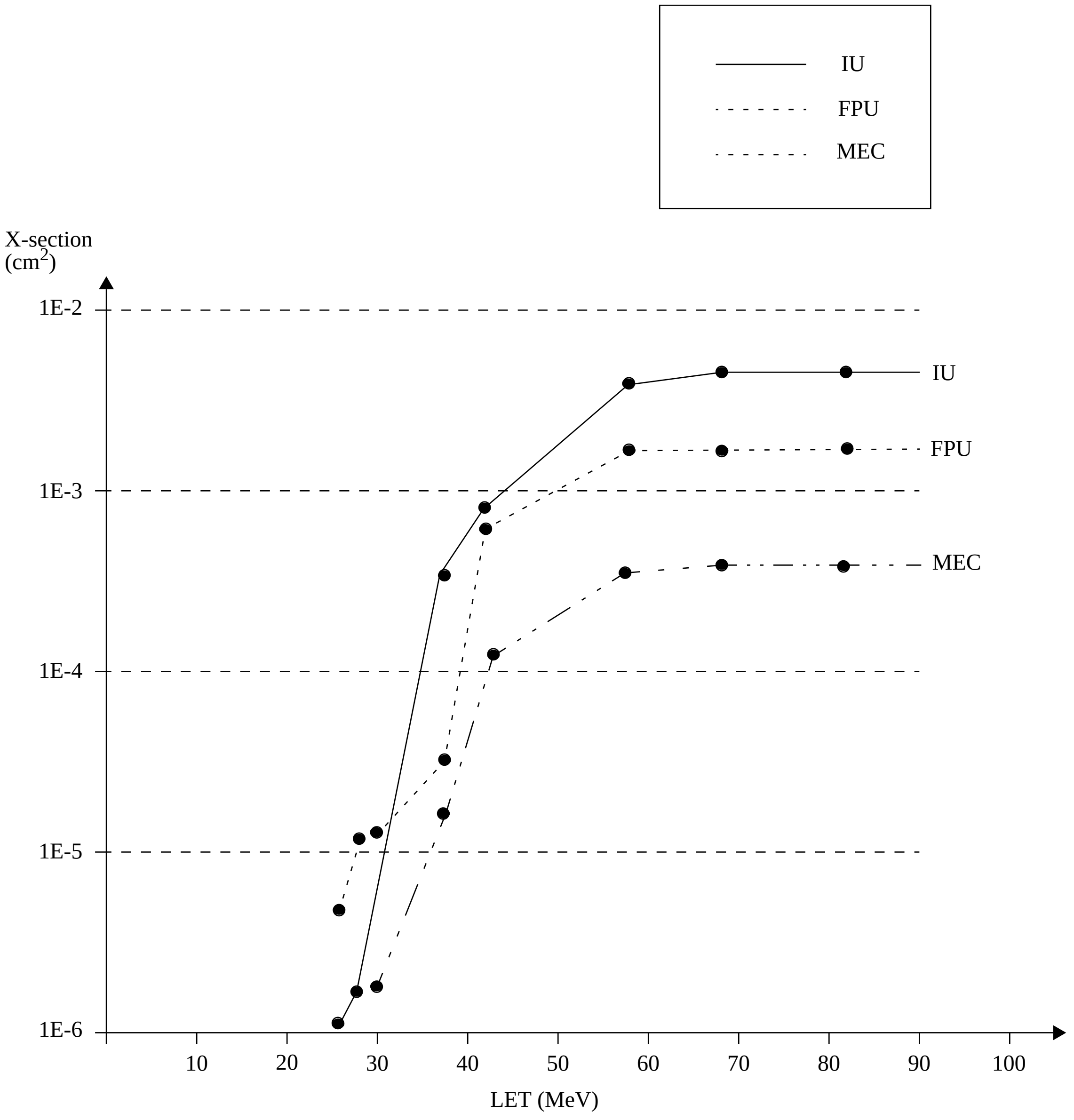

Figure 4: SEU sensitivity for ERC32

4.2 IU error analysis

Table 5 and 6 show the IU test results using the *iureg* program. The *recoverable errors* correspond to IU traps 0x61 and 0x63. O*ther traps* indicate IU traps 0x62, 0x64 and IU error mode (reset trap). *System errors* indicate errors detected by the MEC (bus parity error etc.). *Undetected errors* refer to master/checker errors, software detected failures were never reported during the IU and FPU test due to the usage of a checker device.

Test	Recov. trap	Regfile trap	Other traps	System errors	Detected errors in IU	Detected errors total	Undetec errors	Total errors
Ni-58, 0^o	9	6	2	8	17	25	4	29
Ni-58, 15^o	4	1	0	0	5	5	0	5
Ni-58, 30^o	3	5	3	4	11	15	4	19
Br-79, 0^o	3	402	14	2	419	421	4	425
N-58, 45^o	1	96	2	1	99	100	1	101
Br-79, 30^o	2	343	15	5	360	365	2	367
I-127, 0^o	36	1,294	183	18	1,513	1,531	10	1,541
I-127, 30^o	21	833	140	10	994	1.004	2	1,006
Au-197, 0^o	9	435	61	8	505	513	1	514
TOTAL	**88**	**3,415**	**420**	**56**	**3,923**	**3,979**	**28**	**4,007**

Table 5: IU error distribution

Test	Recov. trap	Regfile trap	Other traps	System errors	Detected errors (IU)	Detected err (total)	Ratio undet	LET	X-sect (cm^2)
Ni-58, 0^o	31%	21%	7%	28%	59%	86%	14%	26.6	1.2E-5
Ni-58, 15^o	80%	20%	0	0	100%	100%	0%	28.5	2.1E-5
Ni-58, 30^o	16%	26%	16%	21%	58%	79%	21%	30.7	1.1E-5 (?)
Br-79, 0^o	0.7%	94.6%	3.3%	0.5%	98.6%	99.1%	0.9%	37.2	4.3E-4
N-58, 45^o	1%	95%	2%	1%	98%	99%	1%	37.6	-
Br-79, 30^o	0.5%	93.5%	4.1%	1.4%	98.1%	99.5%	0.5%	42.9	9.9E-4
I-127, 0^o	2.3%	84%	11.9%	1.2%	98.2%	99.4%	0.6%	58.8	3.7E-3
I-127, 30^o	2.1%	82.8%	13.9%	1%	98.8%	99.8%	0.2%	67.9	4.2E-3
Au-197, 0^o	1.8%	84.6%	11.9	1.6%	98.2%	99.8%	0.2%	82.3	4.3E-3
AVERAGE	**2.1%**	**85.2%**	**10.5%**	**1.4%**	**97.9%**	**99.3%**	**0.7%**		

Table 6: IU relative error distribution

The results from the IU tests shows that the error-detection mechanisms are as efficient as projected; 97.9% of all errors are detected directly by the IU and 99.3% are detected in the system. It can further be noted the register file errors account for more than 85% of all errors, and only 2% of the errors are removable without software intervention. It is estimated that 94% of all on-chip registers were covered by the IU test program.

4.3 FPU error analysis

Table 7 and 8 show the FPU test results using the *fpureg* program. The *recoverable trap* correspond to FPU trap type 6, *non-recoverable trap* to trap type 7 and *data bus trap* to trap type 5.

Test	Recov. trap	Non-recov. trap	Data bus trap	Detected system errors	Detected errors in FPU	Detected errors (total)	Undetec. errors	Total errors	Ratio undetec.
Ni-58, 0^o	9	7	0	0	16	16	1	17	6.2%
Ni-58, 15^o	5	7	2	1	14	15	1	16	6.2%
Ni-58, 30^o	27	6	0	1	33	34	7	41	17%
Br-79, 0^o	20	148	1	1	169	170	10	180	5.6%
Br-79, 30^o	26	211	1	0	237	237	12	249	4.8%
I-127, 0^o	190	866	5	0	1,061	1,061	64	1,125	5.7%
i-127, 30^o	58	410	1	0	469	469	30	499	6.0%
Au-197, 0^o	37	178	3	0	218	218	10	228	4.4%
TOTAL	**372**	**1,833**	**13**	**3**	**2,217**	**2,220**	**135**	**2,355**	**5.7%**

Table 7: FPU error distribution

Test	Recov. trap	Non-recov. trap	Data bus trap	Detected system errors	Detected errors in FPU	Detected errors (total)	Ratio undetec.	LET	X-sect (cm^2)
Ni-58, 0^o	53%	41%	0	0	94%	94%	6.2%	26.6	5.9E-6
Ni-58, 15^o	31%	44%	12%	6%	87%	93.8%	6.2%	28.5	1.3E-5
Ni-58, 30^o	66%	15%	0	2.4%	80%	83%	17%	30.7	1.4E-5
Br-79, 0^o	11%	82%	0.6%	0.6%	93.9%	94.4%	5.6%	37.2	3.8E-4
Br-79, 30^o	10%	85%	0.4%	0	95.5%	95.1%	4.8%	42.9	7.1E-4
I-127, 0^o	17%	77%	0.4%	0	94.3%	94.3%	5.7%	58.8	1.9E-3
i-127, 30^o	12%	82%	0.2%	0	94%	94%	6.0%	67.9	1.8E-3
Au-197, 0^o	16%	78%	1.3%	0	95.6%	95.6%	4.4%	82.3	1.9E-3
TOTAL	**15.8%**	**77.8%**	**0.5%**	**0.1%**	**94.1%**	**94.2%**	**5.8%**		

Table 8: FPU relative error distribution

Table 8 shows that the FPU error-detection coverage is about 94%, somewhat lower than expected. Register file errors account for 77.8% of all errors while 15.8% of all errors are removable without software intervention. The reason for the low error-detection coverage is further discussed below. It is estimated that the FPU test program reached 100% test coverage of all on-chip FPU registers.

4.4 MEC error analysis

Table 9 and 10 show the MEC test results using the *paranoia* program. The *error manager* column correspond to MEC register parity errors detected by the MEC error manager. The *system error* column reflects watchdog time-out, unexpected traps and unexpected resets. *Undetected errors* indicates program failure, i.e. *paranoia* error.

Test	Error manager	System errors	Detected errors	Undetec. errors	Total errors	Ratio undetec.
Ni-58, 0^o	1	6	7	1	8	12%
Ni-58, 30^o	0	8	8	0	8	0%
Br-79, 0^o	6	1	7	0	7	0%
Br-79,3 0^o	19	3	22	0	22	0%
I-127, 0^o	92	15	107	0	107	0%
I-127,3 0^o	86	11	97	2	99	2%
Au-197, 0^o	72	11	83	0	83	0%
TOTAL	**276**	**55**	**331**	**3**	**334**	**0.9%**

Table 9: MEC absolute error distribution

Test	Error manager	System errors	Detected errors	Ratio undetec.	LET	X-sect (cm^2)
Ni-58, 0^o	12.5%	75%	87.5%	12.5%	26.6	7.1E-7
Ni-58, 30^o	0	100%	100%	0%	30.7	2.5E-6
Br-79, 0^o	85.7%	14.3%	100%	0%	37.2	1.8E-5
Br-79,3 0^o	86.3%	13.6%	100%	0%	42.9	1.1E-4
I-127, 0^o	86.0%	14.0%	100%	0%	58.8	4.2E-4
I-127,3 0^o	86.9%	11.1%	98.0%	2.0%	67.9	4.9E-4
Au-197, 0^o	86.7%	13.2%	100%	0%	82.3	4.7E-4
TOTAL	**82.6%**	**16.5%**	**99.1%**	**0.9%**		

Table 10: MEC relative error distribution

Table 10 shows that the MEC overall error-detection coverage is 99.1%. About 82% of the errors were detected directly by the error manager and originated from register parity errors, while 16% were detected through secondary effects. The MEC registers are continuously checked for parity errors by hardware, and the test coverage is therefore independent of the application.

4.5 SEU errors in combinatorial logic

The overall test results are considered to be very promising; 6,530 of 6,696 injected errors were detected corresponding to a system level error-detection coverage of 97.5%. One remaining question is however why the error-detection coverage for the FPU is lower than expected. The FPU has 100% parity protection and should show higher detection coverage. By studying the error distribution at different LET levels, it is apparent that for LET < 37 MeV, register parity errors are less frequent than at LET levels above 37 MeV. For the IU, only 23% of the errors below 37 Mev come from the register file while 85% of all errors above 37 MeV are register file errors. The ratio of recoverable errors, i.e. errors in the first three pipeline stages, is 31% below 37 MeV but only 2% above. The ratio of undetected errors and errors detected at system level, i.e. not on-chip, is also significantly higher below 37 MeV. The FPU and MEC show similar behaviour; the ratio for FPU register file errors is 27% below 37 MeV but 78% above. The recoverable error ration in the FPU is 55% below 37 MeV but only 14% above. For the MEC, only 6% (!) of the errors below 37 MeV are detected through register parity checking, compared to 82% above.

To explain the above behaviour, a hypothesis is made on how SEU errors occur. In addition to a direct hit in a register flipping a bit, an SEU error is assumed to also be able to originate from a hit in combinatorial logic, creating a pulse that is clocked into a register at the clock edge. If the LET threshold for SEU errors in combinatorial logic is lower that the LET threshold for registers, the above described behaviour would be obtained. The hypothesis is supported by following observations:

- The majority of combinatorial logic in the IU is in the three first pipeline stages.
- The FPU has significantly more combinatorial logic than the other two devices.
- The majority of the combinatorial logic in the FPU is in the datapath where a detected error would be recoverable.
- The low on-chip error-detection coverage below 37 MeV in the MEC.

If a SEU error originating from combinatorial logic is detected by register parity depends on the length of the induced pulse and the delay through the parity generator. Typically, parity for a certain data is generated by a parity tree and latched on the same clock edge as the corresponding data. If the delay through the parity generator is longer then the pulse length of the SEU error, the error will be detected since the latched parity bit will not reflect the erroneous data. If the delay in the parity generator is shorter then the SEU pulse length, there is a chance that the erroneous data and erroneous parity bit are simultaneously latched and the error will not be detected. This would explain the low error-detection coverage in the MEC below 37 MeV - the parity in the MEC is mostly built over a few bits only, resulting in a short delay through the parity generator and therefore a higher probability for undetected parity errors. There is also the case where one SEU pulse splits into several pulses creating a “shower” of errors at the input of a register - this can happen if a complex combinatorial structure such as a barrel shifter or multiplier is hit. In this case, it is impossible to predict whether the error will be detected or not.

The effect of SEU in combinatorial logic can be easily checked with new tests, by varying the system clock frequency. The error rate originating from SEU induced errors in combinatorial logic should be proportional to the clock frequency while errors from direct hits in registers occur with the same rate regardless of clock frequency.

5 Absolute error rates

Error calculations for the three devices has been done using CREME software from Naval Research Laboratory. The results are worst-case results assuming solar minimum conditions and a minimum shielding of 1 g/cm^2, but exclude protons and solar flares. The upset contribution from proton-induced nuclear reactions has not been calculated since this requires proton test data. Although it is often assumed that protons are not a problem for devices with sufficiently high LET threshold, this is not always the case and orbits through the proton belt may show an increased upset rate. The upset rate has been calculated for two orbits:

1. GEO, altitude 35,786 km. No geomagnetic shielding - also applies to eccentric orbits.
2. LEO, altitude 400 km, inclination 53^o (space station). Geomagnetic shielding.

Other orbits like HEO or GTO can be expected to give results very similar (but slightly lower) to GEO. The table below shows the upset rate and corresponding FITS figure.

Device	LEO (upset/day)	FITS for all upset errors	Error-detection coverage	FITS for undetected errors	MTBF for all upset errors (years)	MTBF for undetected errors (years)
IU	9.9E-5	4,125	99.3%	29	27.7	3,953
FPU	5.1E-5	2,125	94.2%	123	53.7	926
MEC	9.3E-6	388	99.1%	4	295	32,373
IU + FPU + MEC	**1.6E-4**	**6,638**		**156**	**17.1**	**732**

Table 11: Error rates in LEO

Device	GEO (upset/day)	FITS for all upset errors	Error-detection coverage	FITS for undetected errors	MTBF for all upset errors (years)	MTBF for undetected errors (years)
IU	8.3E-4	34,583	99.3%	242	3.3	471
FPU	4.3E-4	17,917	94.2%	1,039	6.4	110
MEC	8.1E-5	3,375	99.1%	30	33.8	3,758
IU + FPU + MEC	**1.3E-3**			**1,311**	**2.1**	**87**

Table 12: Error rates in GEO

The results show that in LEO, the SEU error-rate for the ERC32 core is approximately one error in 17 years. Taking into account the error-detection mechanisms, the rate for undetected errors is one in 732 years. For GEO, the figures are 2.1 and 87 years respectively.

It should be noted that the above error-rates are absolutely worst-case figures which will not occur in real applications. The test programs used to obtain the error-rates continuously uses 94% - 100% of all registers on both IU and FPU. Analysis from existing programs have shown that only 5 of the 8 IU register windows are used on average, and only half of the registers in each window contains ‘live’ data. This means that the error-rate coming from register-file errors would drop with 69% since errors are only detected when the register value is used. From table 6, it can be seen that register-file errors contribute to 82-95% of all IU errors. It is then a valid assumption that the error-rate for detected errors would be about 50% of the worst-case error-rate. The error-rate for undetected IU errors is not likely to change, since the undetected errors do not originate from hits in the register-file, but rather in unprotected state registers and combinatorial logic. Some reduction in the rate of undetected errors would probably come from the fact that the utilisation of the processor (IU) is seldom 100%, and the idle task is likely to put the processor in power-down mode, i.e. shut down the clock. During this time, hits in combinatorial logic would not lead to any errors. The ratio of undetected errors in the IU is however rather small (0.7%) so this effect can be neglected.

Similar analysis show that even rather floating-point intensive applications only use the FPU 10-20% of the time, with and average of 8 ‘live’ registers. From table 7, it can be seen that FPU register-file errors contribute to about 80% of the error-rate and that 15% of all errors are recoverable without software intervention. With these assumptions the actual FPU rate for detected errors will be less than 5% of the worst-case figure. The rate for undetected errors in not affected by the register usage, but by the FPU utilisation. The rate for undetected errors is the proportional to the FPU utilisation, here assumed to 15% of the worst-case figure. The error rate in the MEC is affected along similar lines as the FPU, but since the error-rate is so small, this effect can be neglected. The tables below show the predicted error-rates adjusted to an average application as described above:

Device	LEO (upset/day)	FITS for all upset errors	Error-detection coverage	FITS for undetected errors	MTBF for all upset errors (years)	MTBF for undetected errors (years)
IU	4.4-E-5	2,062	99.3%	11	55.4	7.906
FPU	2.5 E-6	106	94.2%	6	1,074	18,520
MEC	9.3E-6	388	99.1%	4	295	32,373
IU + FPU + MEC	**1.6E-4**	**2,556**		**21**	**44.7**	**5,436**

Table 13: Error rates in LEO (compensated)

Device	GEO (upset/day)	FITS for all upset errors	Error-detection coverage	FITS for undetected errors	MTBF for all upset errors (years)	MTBF for undetected errors (years)
IU	4.1E-4	17,292	99.3%	121	6.6	942
FPU	2.2E-5	896	94.2%	156	127	732
MEC	8.1E-5	3,375	99.1%	30	33.8	3,758
IU + FPU + MEC	**1.3E-3**	**21,563**		**307**	**5.3**	**372**

Table 14: Error rates in GEO (compensated)

APPENDIX A

Below are some test logs from the SEU monitoring software:

```
! IU test started at Fri Aug 30 12:08:14 1996
M  841396098 Fri Aug 30 12:08:18 1996
DF0  841396112 Fri Aug 30 12:08:32 1996
RS  841396112 Fri Aug 30 12:08:32 1996
RE  841396141 Fri Aug 30 12:09:01 1996
T63  841396149 Fri Aug 30 12:09:09 1996
RS  841396149 Fri Aug 30 12:09:09 1996
RW  841396156 Fri Aug 30 12:09:16 1996
RE  841396173 Fri Aug 30 12:09:33 1996
T63  841396187 Fri Aug 30 12:09:47 1996
RS  841396187 Fri Aug 30 12:09:47 1996
RP  841396323 Fri Aug 30 12:12:03 1996
M  841396323 Fri Aug 30 12:12:03 1996
!  IU error traps    0x61  0x62  0x63  0x64  0x65
!                       0     0     2     0     0
!  Traps  PowRst   SwRst  ErrRst   WdRst    Pass    Fail   McErr
!      0       1       3       2       1     457       0       0
!  FPU error traps (ftt)    0x5   0x6   0x7
!                             0     0     0
!  Detected errors   :     6
!  Undetected errors :     0
!  Total errors      :     6
!  Ratio undetected  :     0%
!  Anomalies         :     0

! Test ended at Fri Aug 30 12:12:23 1996

! FPU test started at Fri Aug 30 11:15:29 1996
M  841392933 Fri Aug 30 11:15:33 1996
TF6  841392956 Fri Aug 30 11:15:56 1996
RS  841392956 Fri Aug 30 11:15:56 1996
TF6  841392961 Fri Aug 30 11:16:01 1996
RS  841392961 Fri Aug 30 11:16:01 1996
I08  841392962 Fri Aug 30 11:16:02 1996
RS  841392962 Fri Aug 30 11:16:02 1996
TF7  841392995 Fri Aug 30 11:16:35 1996
RS  841392995 Fri Aug 30 11:16:35 1996
TF6  841393001 Fri Aug 30 11:16:41 1996
RS  841393001 Fri Aug 30 11:16:41 1996
I08  841393026 Fri Aug 30 11:17:06 1996
RS  841393026 Fri Aug 30 11:17:06 1996
DF0  841393128 Fri Aug 30 11:18:48 1996
!  IU error traps    0x61  0x62  0x63  0x64  0x65
!                       0     0     0     0     0
!  Traps  PowRst   SwRst  ErrRst   WdRst    Pass    Fail   McErr
!      0       0       6       0       0     509       0       0
!  FPU error traps (ftt)    0x5   0x6   0x7
!                             0     3     3
!  Detected errors   :     5
!  Undetected errors :     0
!  Total errors      :     7
!  Ratio undetected  :     0%
!  Anomalies         :     0
! Test ended at Fri Aug 30 11:18:50 1996
! PARANOIA test started at Fri Aug 30 17:41:35 1996
M  841416099 Fri Aug 30 17:41:39 1996
RE  841416360 Fri Aug 30 17:46:00 1996
RE  841416362 Fri Aug 30 17:46:02 1996
RE  841416382 Fri Aug 30 17:46:22 1996
RE  841416612 Fri Aug 30 17:50:12 1996
RE  841416667 Fri Aug 30 17:51:07 1996
RW  841416868 Fri Aug 30 17:54:28 1996
RE  841416922 Fri Aug 30 17:55:22 1996
!  IU error traps    0x61  0x62  0x63  0x64  0x65
!                       0     0     0     0     0
!  Traps  PowRst   SwRst  ErrRst   WdRst    Pass    Fail   McErr
!      0       0       0       6       1    3670       0       0
!  FPU error traps (ftt)    0x5   0x6   0x7
!                             0     0     0
!  Detected errors   :     7
!  Undetected errors :     0
!  Total errors      :     7
!  Ratio undetected  :     0%
!  Anomalies         :     0
! Test ended at Fri Aug 30 17:56:44 1996
```