一种针对软错误的程序可靠性定量分析方法

徐建军,谭庆平,熊 磊,叶 俊
(国防科学技术大学计算机学院,湖南长沙 410073)

摘 要: 宇宙射线辐射所导致的软错误是航天计算面临的最主要挑战之一.而随着集成电路制造工艺的持续进步,现代处理器的计算可信性日益面临着软错误的严重威胁.当前,很少有研究从软件角度分析软错误对系统可靠性的影响.在程序汇编代码的基础上,提出一种针对软错误的程序可靠性定量分析方法 PRASE,并且提出基本块分析法和 3 条运算定律以改进其分析效率.实验表明软错误对程序可靠性的影响与程序自身结构密切相关,同时分析结果还指出在软错误影响下程序的关键脆弱点,对实现针对软错误的高效容错算法有参考意义.

关键词: 软错误;单粒子翻转;程序可靠性分析;错误传播
中图分类号: TP311 **文献标识码:** A **文章编号:** 0372-2112 (2011) 03-0675-05

A Quantitative Approach for Program Reliability Analysis of Soft Errors

XU Jian-jun, TAN Qing-ping, XIONG Lei, YE Jun
(*School of Computer, National University of Defense Technology, Changsha, Hunan* 410073, *China*)

Abstract: Soft errors, caused by the radiation of cosmic rays, are one of the top challenges for space computing. With the continuously progress of integrated circuits, the dependability of modern processors is increasingly affected by soft errors. Currently, few works analyze the impact of soft errors from the perspective of software. This paper presents a quantitative approach, named PRASE, which is able to analyze the effect of soft errors to the reliability of a given program. Moreover, the analytical performance is increased significantly with the help of the basic block analysis and three arithmetic laws. The experiments show that the reliability of a program has a connection with its native structure. Furthermore, the analytical results also identify the vulnerabilities of a given program under the occurrence of soft errors. These contributions are in favor of implementing the high efficient algorithms for tolerating soft errors.

Key words: soft error; single event upset; program reliability analysis; error propagation

1 引言

软错误(Soft Error)[1,2]是半导体电路的一种瞬态故障现象,通常由外部环境中的高能粒子辐照和电磁干扰等电子噪声诱发.当高能粒子轰击半导体电路时,会导致 PN 结出现瞬间充放电,从而改变内部逻辑状态,例如从逻辑 0 变成逻辑 1,这种现象也被称为单粒子翻转(Single Event Upset,简称 SEU).

大气层外的电子设备处于由大量高能带电粒子所构成的强辐射环境中,所以 SEU 是航天计算的最主要挑战之一.而随着处理器逐步采用深亚微米制造工艺,在性能得到大幅提高的同时,对于引起软错误的各种噪声干扰也变得越来越敏感[2].芯片集成晶体管数目呈指数级增长,使整体软错误率(Soft Error Rate,简称 SER)快速增加.文献[3]预测处理器 SER 将会以每代 8% 的速度递增.继性能和功耗之后,软错误所导致的计算可信性问题已成为一个日益严峻的课题.

当前关于软错误的分析研究主要在硬件层上进行,已提出多种针对硬件可靠性的分析技术[4,5].但是这些研究没有针对所运行程序进行分析,不能得出软错误对系统可靠性影响的完整结果.程序计算导致的信息交互作用会发生错误传播,软错误的特点决定了必须同时考虑软件的因素.而且从软件角度进行分析有助于理解程序自身结构与软错误对程序可靠性影响之间的关系,明确其中的关键脆弱点,从而有利于采取针对性的错误检测和恢复技术,对研究针对软错误的程序编译技术和设计方法也有参考意义.

传统的软件可靠性研究通常假设运行在理想硬件上[6],不适合分析由软错误所带来的可靠性问题.在概率理论和程序汇编代码的基础上,本文提出一种针对软错误的程序可靠性定量分析方法 PRASE(Program Reliability Analysis for Soft Errors).对于程序中的数据和指令,

收稿日期:2009-11-11;修回日期:2010-11-28

PRASE 根据其在存储单元中的驻留时间计算原始软错误的产生概率,并基于基本错误集合分析错误传播过程.为改进 PRASE 的分析效率,还提出基于基本块的加速分析方法和 3 条运算定律.

2 PRASE 基本模型

在程序汇编代码的基础上,PRASE 目前关注发生在存储部件(如内存和寄存器)中的软错误.把一条指令定义为 $insn=(S,d,\#n)$,其中 $S=\{s_1,s_2,\cdots,s_k\}$ 为源操作数集合,d 为目的操作数,$\#n$ 为指令序列号.这里不考虑流水线等底层硬件,设指令依次顺序执行,每条指令在一个指令周期内完成执行.

2.1 错误产生模型

PRASE 将存储具体指令和数据的存储块称为基本单元,并分为 3 类:①代码,用指令序列号标识,如 #1;②数据,用所处地址加前缀'&'标识,如 &12;③寄存器,用寄存器名标识,如 $2(存储控制流信息的程序计数器标识为 pc).基本单元集合记为 $UNIT$.

对于基本单元 u,在时间 t 内 u 保持正确和出现软错误的概率记为 $P_s(u)$ 和 $P_f(u)$,显然 $P_s(u)+P_f(u)=1$.通常认为一个二进制位 b 的原始 SER 随时间服从指数分布.设原始 SER 为 λ,那么在时间 t 内 b 出现软错误的概率 $P_f(b)=1-e^{-\lambda t}$,则 $P_s(b)=e^{-\lambda t}$.对于 u 来说,如果 u 任一个二进制位出现软错误,则 u 中数据被破坏,即 u 出错.易知若事件 A 和 B 是相互独立的,则逆事件 $\overline{A}$ 和 $\overline{B}$ 也相互独立.PRASE 采用 SEU 软错误模型,表示 u 中的每个二进制位是否发生软错误相互独立,所以对应保持正确的概率也相互独立.可得:

$$P_s(u)=P_s(b_1\cdot b_2\cdot\cdots\cdot b_l)=\prod_{i=1}^{l}P_s(b_i)=\prod_{i=1}^{l}e^{-\lambda t}=e^{-\lambda tl} \tag{1}$$

$$P_f(u)=1-P_s(u)=1-e^{-\lambda tl} \tag{2}$$

λ 值取决于存储部件的电路特征和工作环境,如工作电压、高能粒子辐照能量等,可通过一些模拟方法确定 λ 值[7].时间 t 表示数据在 u 中的驻留时间.l 是 u 的有效二进制位数(记为 $L(u)$),与具体指令集有关.将连续两次读(或写后读)的间隔时间内发生软错误的事件称为基本错误[5].图 1 示例了 PRASE 分析时基本错误的产生过程,每次读取都产生一个基本错误,但是写入时出错情况只取决于写入的数据.将 t 时刻由于读取 u 而产生的基本错误记为 $\varepsilon_{u,t}$.

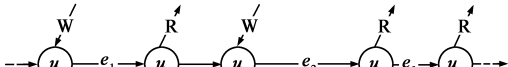

图1 基本错误产生示例

用 $T(u)$ 表示 u 从最近一次访问到当前时刻所经历的时间.用 $\phi(\varepsilon_{u,t})$ 表示基本错误 $\varepsilon_{u,t}$ 的效应值,即 $\phi(\varepsilon_{u,t})=T(u)\times L(u)$.根据等式(2)和函数 ϕ 的定义,$\varepsilon_{u,t}$ 发生概率 $P(\varepsilon_{u,t})=1-e^{-\lambda\phi(\varepsilon_{u,t})}$.在 SEU 模型中,不同基本单元是否发生软错误相互独立,因此基本错误之间相互独立.

2.2 错误传播模型

错误传播模型的目标是确定受错误传播影响数据出错的概率.PRASE 认为一条指令的正确执行有三个条件:①源操作数正确;②指令代码没有被软错误篡改;③指令按正确顺序被执行.需要指出这是一种保守假设,但例外情况出现的概率非常小,PRASE 期望求得软错误对可靠性影响的上界值,所以要求满足这三个条件.对于指令目的操作数 d 有:

$$P_s(d)=P_s(s_1\cdot s_2\cdot\cdots\cdot s_k\cdot\#n\cdot pc) \tag{3}$$

在实际程序中,等式(3)中基本单元出错的概率很多时候是相关的,使得难以直接计算 $P_s(d)$.为解决该问题,PRASE 采用文献[5]提出的思想——不直接计算错误发生的概率,而是跟踪错误产生和传播的过程.为每个基本单元定义一个基本错误集合 $E:UNIT\to P(ERROR)$,$E(u)$ 包含了传播到 u 和直接发生在 u 的基本错误.对于一条指令,其源操作数的基本错误集合会因读取操作而增加新的元素,目的操作数的基本错误集合依赖于指令的源操作数、指令代码和 pc.根据等式(3)的分析得:$E(d)=E(s_1)\cup E(s_2)\cup\cdots\cup E(s_k)\cup E(\#n)\cup E(pc)=\{e_1,e_2,\cdots e_m\}$.由基本错误的独立性求得关于等式(3)的进一步结果:

$$P_s(d)=P(\overline{E(d)})=P(\overline{e_1}\cdot\overline{e_2}\cdot\cdots\cdot\overline{e_m})=\prod_{i=1}^{m}P(\overline{e_i})$$

$$=\prod_{i=1}^{m}(1-P(e_i))=e^{-\lambda(\phi(e_1)+\phi(e_2)+\cdots+\phi(e_m))} \tag{4}$$

定义 1 软错误影响下程序状态 $state=(\tau,BE,T,E)$:

- τ,表示程序执行到当前指令所经历的时间;
- BE,当前时刻分析产生的基本错误的集合;
- $T:UNIT\to Z^+\cup\{0\}$;
- $E:UNIT\to P(BE)$.

初始状态 S_{init} 中,$\tau=0$,$BE=\varnothing$,$\forall u\in UNIT:T(u)=0\wedge E(u)=\varnothing$.对于指令 $insn=(S,d,\#n)$,设分析前后程序状态分别为 $S_{\bigcirc}$ 和 $S_{\bullet}$,按上述讨论得转化方程组如下:

$$\begin{cases}\tau_{\bullet}=\tau_{\bigcirc}+1\\ BE_{\bullet}=BE_{\bigcirc}\cup\{\varepsilon_{u,\tau_{\bullet}}\mid\phi(\varepsilon_{u,\tau_{\bullet}})=T_{\bigcirc}(u)\times L(u),u\in S\cup\{\#n,pc\}\}\\ T_{\bullet}(u)=\begin{cases}0 & ,u\in S\cup\{\#n,pc,d\}\\ T_{\bigcirc}(u)+1 & ,\text{otherwise}\end{cases}\\ E_{\bullet}(u)=\begin{cases}E_{\bigcirc}(u)\cup\{\varepsilon_{u,\tau_{\bullet}}\} & ,u\in S\cup\{\#n,pc\}\\ \bigcup(E_{\bullet}(v)\mid v\in S\cup\{\#n,pc\}) & ,u=d\\ E_{\bigcirc}(u) & ,\text{otherwise}\end{cases}\end{cases}$$

2.3 程序可靠性计算

根据程序指令执行序列,按上述方法可通过逐条指令分析的方式求程序最终状态.在最终状态中,PRASE关注输出结果对应的基本单元,设这些基本单元的集合为 $O=\{o_1,o_2,\cdots,o_n\}$(程序控制流不能被破坏,该集合包含 pc).如果 $E(O)$有 k 个基本错误,程序一次执行的可靠性为:

$$P_s(O)=\mathrm{e}^{-\lambda(\phi(e_1)+\phi(e_2)+\cdots+\phi(e_k))} \tag{5}$$

程序会有不同的执行路径,而不同路径的错误产生和传播过程各不相同,PRASE分析需要在程序执行路径分析的结果基础上进行.假设已有程序执行路径 $path_1,path_2,\cdots,path_n$,以及它们的执行概率(记为 $\rho_1,\rho_2,\cdots,\rho_n$).对程序整体可靠性来说有:

$$P_s(prog)=\sum_{i=1}^{n}(\rho_i\times P_s(O_i)) \tag{6}$$

把一条路径出现的指令数作为路径执行时间,记为 $C(path_i)$.假设程序循环运行,可得下列最终结果:

$$P_f(prog)=1-P_s(prog) \tag{7}$$

$$C(prog)=\sum_{i=1}^{n}(\rho_i\times C(path_i)) \tag{8}$$

$$\mathrm{MTTF}=C(prog)/P_f(prog) \tag{9}$$

其中,$P_f(prog)$和 $C(prog)$分别表示程序一次执行的平均失效概率和执行时间,MTTF表示程序的平均无故障时间.

3 分析优化方法

逐条指令分析可求得最终结果,但 BE 中的基本错误数会快速增长,对于实际程序还不可行.为此,PRASE提出一种基于基本块的优化方法.

基本块是能够被顺序执行的指令最大集合.基于基本块,程序可表示为一个由基本块构成的控制流图,每条执行路径都可被映射到控制流图上.因为基本块的指令执行序列固定,如果其中每个基本单元确定,则其内部的错误产生和传播过程也能被确定.PRASE预先分析基本块的错误产生和传播过程,然后结果被整体程序分析所使用.先为基本块定义数据依赖图(Data Dependency Graph,简称 DDG).

定义 2 对于一个基本块 B,其 $DDG=(\tau_b,U_b,V,L,T_b,BE_b,W)$,其中:

– τ_b,表示 B 执行所需时间,即 B 包含的指令数.

– U_b,出现在 B 中的基本单元的集合.

– V,DDG 顶点集合,表示 B 中的基本单元实例.V 有两个特殊子集,V_{entry}和 V_{exit},分别代表初始和最终的基本单元实例集合.顶点 v 的对应基本单元记为 $\delta(v)$.

– $L\subset V\times V$,表示 DDG 的边集合,被分成 L_{gen}和 L_{prop}两个部分,分别表示错误产生和错误传播关系.

– $T_b:V\to Z^+\cup\{0\}$,表示基本单元实例相对于基本块入口处被访问的时刻.

– $BE_b=\{\varepsilon_{v,T_b(v)}\mid v\in V\}$,是产生于 B 中的基本错误集合.BE_b 有一个特殊的子集 $BE_?$,其中对应的基本单元上一次访问在 B 执行之前,所以实际效应值还依赖于 B 之前的状态.

– $W\subseteq L_{\mathrm{gen}}\to BE_b$,表示 L_{gen}中边 l 对应的基本错误,其驻留时间取决于两端点的访问时间间隔.

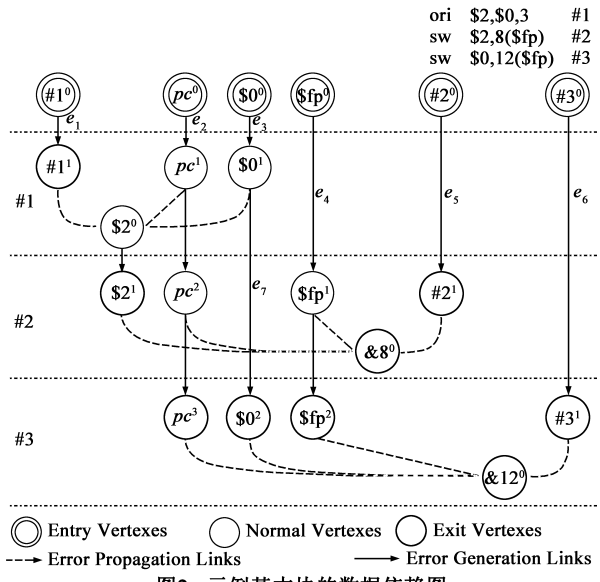

图2 示例基本块的数据依赖图

图2是右上角示例基本块的 DDG.基本单元的实例通过上标进行区分,L_{gen}和 L_{prop}中的边分别用实线和虚线表示.有7个基本错误被生成,其中 $e_7\notin BE_?$,间隔小于一个周期的基本错误被忽略,如 $pc^1\to pc^2$.

对于 U_b 中的基本单元 u,可根据 u 在 V_{exit}中的对应顶点生成一颗错误传播树.基于该树,可为 U_b 中基本单元定义在该基本块的基本错误集合 E_b:$U_b\to P(\{\cup\{E(\delta(v))\mid v\in V_{\mathrm{entry}}\}\}\cup BE_b)$.例如,$E_b(\$2)=\{e_1,e_2,e_3\}\cup E(\#1)(E(pc)\cup E(\$0)$.

当基本块之前状态 $S_{\bigcirc}$确定后,先要根据 DDG 中的 BE_b 产生基本错误实例(基本错误 e 的实例记为 $\eta(e)$).然后,基本块之后状态 $S_{\bullet}$按下面方程组求得:

$$\begin{cases}\tau_{\bullet}=\tau_{\bigcirc}+\tau_b\\ BE_{\bullet}=BE_{\bigcirc}\cup\{\eta(e)\mid e\in BE_b\}\\ T_{\bullet}(u)=\begin{cases}\tau_b-T_b(v), & u=\delta(v)\wedge v\in V_{\mathrm{exit}}\\ T_{\bigcirc}(u)+\tau_b, & \text{otherwise}\end{cases}\\ E_{\bullet}(u)=\begin{cases}\{\eta(e_i),\cdots,\eta(e_j)\}\\ \cup E_{\bigcirc}(u_m)\cup\cdots\cup E_{\bigcirc}(u_n),\\ E_{\bigcirc}(u)\end{cases}\begin{array}{l}E_b(u)=\{e_i,\cdots,e_j\}\\ \cup E(u_m)\cup\cdots\cup E(u_n)\\ \wedge u=\delta(v)\wedge v\in V_{\mathrm{exit}}\\ \text{otherwise}\end{array}\end{cases}$$

可按分析状态的构成证明与逐条指令分析是等价

的,这里不展开讨论.基本块分析的效率要好很多,因为每条指令只分析一次,然后在整体程序分析只需实例化分析结果,实验发现分析效率被提高了一个数量级以上.但前提是要标识出 U_b 中的基本单元,难点在于如何确定位置动态变化的基本单元(如数组成员对应数据).为解决该问题,PRASE 在分析基本块 *DDG* 时采用符号化方式进行表示,在使用 *DDG* 结果时根据程序执行路径对其中的动态位置实例化.

PRASE 进一步提出下面 3 条运算定律.

PC 合并律 所有传播到 *pc* 的基本错误都可以被合并成一个基本错误.

块组合律 一条执行路径的两个相邻基本块的实例化过程中,如果所有符号表示相同的基本单元实例化后所指向的实际位置相同,则可将这两个块组合成一个更大的块进行分析.

结合律 对于具有公共子路径的执行路径,如果公共子路径在所属路径的实例化过程中,对应位置的基本单元实例化后所指向的实际位置相同,则可将公共子路径单独提取出分析,且公共子路径的执行概率等于所属路径概率之和.

这 3 条运算定律的目的是通过减少基本错误的实例数来提高分析效率,具体证明从略.在实际实验中,使用这 3 条运算定律后 PRASE 分析效率在基本块分析的基础上进一步提高了 4 ~ 6 倍.

4 分析实验

选取了 6 个基准程序进行了分析实验:bsort(冒泡排序)、qsort(快速排序)、hsort(堆排序)、fib(求 Fibonacci 数)、mm(矩阵乘法)和 shuf(随机数测试).先把程序交叉编译成 MIPS32 的汇编代码.

表 1 分析实验结果

Name	C(prog) [cycles]	TotalEffect [kb-cycles]	ValidEffect [kb-cycles]	*PVF* [%]	$P_f(prog)$ [$\times 10^{-13}$]	MTTF [days]
bsort	6,979	23,553	17,860	75.8	10.1	3200.5
hsort	5,526	33,158	24,506	73.9	13.9	1846.9
qsort	3,091	22,104	12,906	58.4	7.3	1961.6
fib	351,763	1,967,674	824,852	41.9	466.3	3492.9
mm	466,253	36,688,283	24,607,476	67.1	13909.3	155.2
shuf	1,310,084	12,445,798	8,014,213	64.4	4530.0	1338.9

分析结果见表 1.第 2 列是程序一次执行的平均执行周期.第 3 列的 TotalEffect 是程序平均执行周期与所有基本单元二进制位数和的乘积,表示在程序执行过程中所有可能被软错误影响的区域.第 4 列的 ValidEffect 反映了会影响程序可靠性的软错误,是 $E(O)$ 中所有基本错误的效应值之和.

第 5 列 *PVF* = (ValidEffect / TotalEffect),具体表征了程序容软错误的能力,值越小则表示程序受软错误的影响越小.为计算程序的可靠性,软错误率(取 5.65×10^{-17}errors/kb-cycle,它相当于在主频为 25MHZ 的处理器中每兆数据一天平均发生 1 次软错误.由于程序规模和 λ 取值较小,使得实验程序一次执行的可靠性都非常高,第 6 列给出程序的失效概率 $P_f(prog)$.

第 7 列的 MTTF 是可靠性分析的最终结果,从该结果至少可以得出两个结论:①程序的时空复杂度越高,软错误产生和传播的机会越大,可靠性越低.如'mm';②*PVF* 值越小,则程序结构越有利于屏蔽软错误,使得软错误对程序可靠性的影响越小,如'fib'.

比较 3 个排序程序可以发现,虽然所实现功能一致,但在程序可靠性和性能方面存在差异.其中快速排序算法的效率和可靠性都是最高的.而'hsort'虽然效率要比'bsort'高,但是可靠性却低于'bsort'.原因是堆排序算法要维持一个堆结构,堆结构被软错误破坏则不到正确的结果,所以堆排序受软错误的影响要高于冒泡排序.因此在有软错误发生的情况下,与只考虑时空复杂度不同,还必须分析程序的容软错误能力.

图 3 给出 3 个程序的基本错误产生过程,横纵坐标分别是程序执行周期和所产生的基本错误效应值的归一化结果,可以发现各个程序存在明显差异.以'fib'为例,求 Fibonacci 数的函数在递归调用自身时,先要把函数当前现场保存下来.如果递归调用层次很深,那么当恢复现场时保存现场的数据可能已经驻留很长时间,此时产生效应值非常大的基本错误.

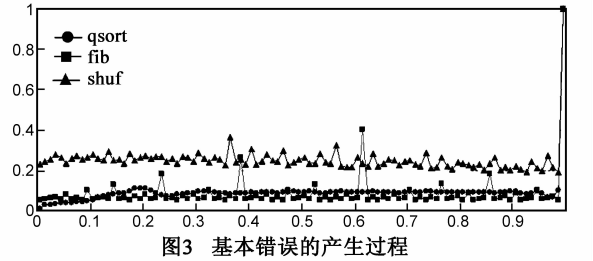

图3 基本错误的产生过程

图 3 反映了这些程序在软错误存在时的关键脆弱点,为设计和选择软错误容错算法提供了依据.如果优先对那些出错概率大的基本单元进行错误检测和恢复,就能够以较低的开销获得程序可靠性的大幅度提升.例如对于'fib',可以优先保护用于保存函数调用现场的数据,并在函数返回时对这些数据进行检测.

5 结论

PRASE 方法能够在汇编代码的基础上定量分析软错误在程序中的产生和传播过程,软错误对程序可靠性的影响基于程序执行剖面求得,基本块分析法和 3 条运算定律显著提高了使得 PRASE 的分析效率.实验表明在软错误发生的情况下,程序可靠性与其本身结构密切相关.PRASE 对硬件进行了简化,分析结果与实际

情况存在差距.但是足以反映程序脆弱点等相对性指标,为设计和实现高效的容错算法提供依据.

参考文献

[1] Baumann R C. Radiation-induced soft errors in advanced semiconductor technologies [J]. IEEE Trans on Device and Materials Reliability, 2005, 5(3): 305 – 316.

[2] Ziegler J F, Puchner H. SER-History, Trends, and Challenges: A Guide for Designing with Memory Ics [M]. San Jose, CA: Cypress Semiconductor Corp, 2004.

[3] Shivakumar P, Kistler M, et al. Modeling the effect of technology trends on the soft error rate of combinational logic [A]. Proc of the Int'l Conf on Dependable Systems and Networks [C]. Bethesda, MD: IEEE Computer Society Press, 2002. 389 – 399.

[4] Mukherjee S S, Weaver C, et al. A systematic methodology to compute the architectural vulnerability factors for a high-performance microprocessor [A]. Proc of the 36th Int'l Symp on Microarchitecture [C]. San Diego, CA: IEEE Computer Society Press, 2003. 29 – 40.

[5] Li X, Adve S V, Bose P, Rivers J A. SoftArch: An architecture-level tool for modeling and analyzing soft errors [A]. Proc of the Int'l Conf on Dependable Systems and Networks [C]. Yokohama, Japan: IEEE Computer Society Press, 2005. 496 – 505.

[6] Littlewood B. Software Reliability: Achievement and Assessment [M]. Oxford, UK: Blackwell Ltd, 1987.

[7] Gordon M S, Goldhagen P, et al. Measurement of the flux and energy spectrum of cosmic-ray induced neutrons on the cround [J]. IEEE Trans on Nuclear Science, 2004, 51(6): 3427 – 3434.

作者简介

徐建军　男,1980 年生,国防科技大学计算机学院博士生,研究方向为程序分析和容错编译.
E-mail: jjun.xu@gmail.com

谭庆平　男,1965 年生,国防科技大学计算机学院教授,博士生导师,研究方向为分布式软件工程和高可信软件.

熊　磊　男,1981 年生,国防科技大学计算机学院博士生,研究方向为软件容错.

叶　俊　男,1982 年生,国防科技大学计算机学院博士生,研究方向为形式化验证.