Extended TED 를 이용한 임베디드 시스템의 데이터 오류 감지 성능 개선

강민구 º, 박기진

아주대학교 산업정보시스템공학부
e-mail : {ozige°, kiejin}@ajou.ac.kr

Improving Data Error Detection Performance for Embedded Systems Using Extended Temporal Error Detection (E-TED)

Minkoo Kang, Kiejin Park

Division of Industrial & Information Systems Engineering, Ajou University

요 약

임베디드 시스템의 신인도(Dependability)를 확보하기 위해 하드웨어적으로 여분을 두는 결함 허용(Fault-tolerant) 기법을 적용하는 것은 임베디드 시스템의 설치공간, 비용 및 전원 공급의 부족 등의 이유로 무리가 있다. 본 논문에서는 소프트웨어 결함 허용 기법의 일종인 시간 여분(Time-redundancy) 개념을 적용한 Extended Temporal Error Detection (E-TED) 기법을 연구하였으며, 실험을 통해 제안한 E-TED 기법이 기존의 TED 기법에 비해 데이터 오류의 감지 확률이 높은 것을 확인하였다.

1. 서론

최근 자동차용 및 산업용 임베디드 시스템은 높은 신인도 (Dependability: Reliability, Availability, Safety, etc)를 요구하고 있으며, 임베디드 시스템의 신인도를 확보하기 위해서는, 원하는 시간 내에 원하는 서비스를 제공할 수 있는 능력을 지속적으로 관리할 수 있어야 한다[1].

일반적으로, 컴퓨터 시스템에 내재하는 결함(Fault)이 미처 제거되지 못하고 발현하게 되면 오류(Error)를 야기하며, 이러한 오류가 지속적으로 발생할 경우, 시스템 고장(Failure) 상태에 이르게 된다. 때문에 시스템 고장을 막기 위해서는 근원적으로 결함을 다루어야 한다. 컴퓨터 시스템에서는 결함 방지(Prevention), 결함 허용(Tolerance), 결함 제거(Removal), 결함 예측(Forecasting)과 같은 대표적인 결함 처리 기술을 사용하고 있으며, 이들 중에서 결함 허용 기술은 오류를 찾고(Detection), 그것을 복구(Recovery)하는 방법을 사용하며, 앞서 말한 4 가지 결함처리 기술 중에서 가장 활발히 연구/응용되는 분야이다[2].

기존 컴퓨터 시스템에서 성공적으로 사용되던 결함처리 기술은 하드웨어/소프트웨어적으로 여분(Redundancy)을 사용하는 방법을 이용하였지만, 이 기술들을 임베디드 시스템의 신인도 확보를 위해 그대로 사용하는 것은 설치 공간, 비용 및 전원 공급의 부족 등의 이유로 무리가 있다고 판단된다. 때문에 임베디드 시스템의 신인도 확보를 위해서는 하드웨어 및 소프트웨어적으로 여분을 가능한 최소한으로 사용하면서, 마이크로프로세서의 처리 능력을 최대한 활용할 수 있는 시간 여분(Time-redundancy) 개념을 적용한 결함 허용 기법을 사용하는 것이 적합하다고 할 수 있다.

시간 여분 기법이 적용된 임베디드 시스템의 데이터 오류를 처리하기 위해서는, 각각의 작업(Task)의 마감시간(Deadline)을 기준으로 적절한 반복 실행(Execution replicas)의 수를 결정하는 것이 중요하며, 또한 결함을 감지할 확률을 구하는 것이 필요하다. 본 논문의 2 장에서는 관련 연구를 언급하고, 3 장에서는 Extended TED 를 정의하였으며, 4 장에서는 데이터 오류를 감지할 확률 식을 정의 하고, 5 장에서는 제안한 방법의 성능 평가를 수행하였으며, 6 장에서는 결론을 내렸다.

2. 관련 연구

컴퓨터 시스템의 결함허용을 위해 시간 여분을 이용하는 방법은 잘 알려져 있는 기술이며[3], 이러한 기술은 주로 데이터의 오류를 감지하기 위해 사용되었다. [4]에서는 하드웨어의 일시적 오류를 감지하여

"본 연구는 2005년 정부(교육인적자원부)의 재원으로 한국학술진흥재단의 지원을 받아 수행되었음." (KRF-2005-041-D00630)

데이터의 무결성을 보장함과 동시에 소비되는 자원(Resource)을 최소화하는 소프트웨어 결함 허용 기법을 연구하였으며, 프로시저(Procedure) 레벨에서의 시간 여분 기법을 응용하였다.

TEM(Temporal Error Masking)은 작업의 마감시간에 여유가 있을 경우, 2 회 또는 3 회의 반복 실행과 투표(Voting)과정을 통해 일시적 결함(Transient Fault)에 의해 발현된 오류를 감추는 방법으로, 이러한 TEM 을 이용하여 오류를 감춤(Masking)으로써 노드(Node)의 고장을 막을 수 있다는 연구[5]에서는 Brake-by-wire 응용에 결함 주입(Fault Injection) 실험을 수행함으로써, TEM 을 적용하기 전에 발생되었던 고장의 수를 42% 가까이 줄일 수 있었다.

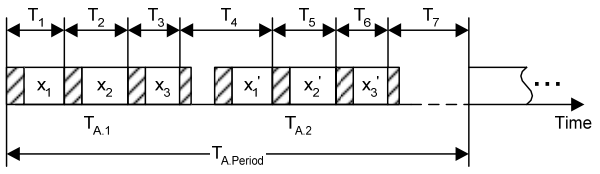

그림 1. TED 를 사용한 작업의 처리 과정

[6]에서는 TED(Temporal Error Detection) 및 TEM 을 이용하여 영구적 결함(Permanent Fault)에 의해 발현된 데이터 오류를 감지할 확률 식을 정의하였으며, 결함 투입 실험을 통해 제안한 확률 식을 검증하였다. 그러나 여분 작업의 실행이 끝나고, 다음의 새로운 작업을 시작하기 전까지의 시간 간격(예를 들어 그림 1 의 T_7, 그림 1 의 각 파라미터에 대한 정의는 3 장을 참조)이 길어질수록 데이터 오류의 감지 확률이 점점 낮아지는 단점이 있다. 이에 본 논문에서는 작업의 마감시간이 허락하는 한도 내에서 반복 실행의 수를 최대화하는 Extended TED(이하 E-TED) 기법을 정의하고, E-TED 기법을 사용하였을 때의 데이터 오류 감지 확률 식을 도출하였다. 또한 실험을 통해 도출된 확률 식을 검증하였으며, E-TED 기법이 기존의 TED 기법에 비해 데이터 오류의 감지 확률이 높은 것을 확인하였다.

3. E-TED 의 정의

본 절에서 영구적 결함에 의해 발현된 데이터 오류를 감지하기 위해 사용한 E-TED 기법에 적용한 가정은 다음과 같다.

A1: 영구적 결함은 작업의 실행 시간 동안 균일 분포(Uniform Distribution)를 따라 발생하며, 하나의 작업 내에서 2 개 이상 발생하지 않는다.
A2: 영구적 결함이 발현할 확률은 작업 내의 모든 산술 연산 과정 동안에 대해 일정하다.
A3: 하나의 작업은 여러 개의 산술 연산(Arithmetic Operations)으로 구성되며, 영구적 결함이 각 산술 연산에서 발현할 확률은 서로 독립이다.
A4: 각각의 산술 연산을 실행하는 시간에 비해 반복 작업들의 결과 값을 비교하는 시간은 매우 작으므로 이는 고려하지 않는다.
A5: 작업의 실행 시간 동안 다른 작업에 의한 인터럽트(Interrupt)는 발생하지 않는다.

E-TED 기법은 각 작업을 반복하여 실행한 뒤 그 결과 값을 비교하는 방법을 말하며, 영구적 결함에 의해 발생한 데이터 오류를 감지하기 위해 사용되었다. 여기에서 반복 작업의 수(e)는 작업의 시작시간(T_s), 마감시간(T_d) 및 1 회 실행시간 (T_e)을 이용하여 정의하였으며, 다음의 수식 (1)과 같다.

$$e = \left\lfloor \frac{T_d - T_s}{T_e} \right\rfloor \tag{1}$$

하나의 작업은 a 개의 산술 연산으로 구성되며, 작업의 마감시간에 이르기 전까지 e 번의 반복 실행을 하게 되므로, 각 작업은 산술 연산을 기준으로 $N(=ea+1)$ 개의 시간 구간으로 나눌 수 있으며, 이 때의 시간 구간들을 T_1 ,..., T_N 이라고 정의하였다. 또한 i 번째 반복 실행의 j 번째 산술 연산은 x_j^i 으로 정의하였다(그림 2 참조).

4. 데이터 오류 감지 확률

영구적 결함이 임의의 시간 구간, T_i 에서 발생할 확률(P_i)은 **A1** 에 의해 다음의 수식 (2)와 같이 정의하였다.

$$P_i = \frac{T_i}{\sum T_i} = \frac{T_i}{T_d - T_s},\ i = 1,2,\cdots,N \tag{2}$$

임의의 산술 연산, x_j^i 에서 영구적 결함이 발현할 확률(Px_j^i)는 **A2** 에 의해 독립적인 일정한 값을 가지며, 다음의 수식 (3)과 같이 정의하였다.

$$Px_j^i = Px,\ i = 1,2,\cdots,e,\ j = 1,2,\cdots,a \tag{3}$$

영구적 결함이 발현함에 따라 작업의 i 번째 반복 실행의 결과 값이 달라질 확률(Px^i)은 i 번째 반복 실행을 구성하는 모든 산술 연산들 중 적어도 하나 이상의 산술 연산에서 영구적 결함이 발현할 확률과 같다. 또한 A3 에 의해 각각의 산술 연산에서 영구적 결함이 발현할 확률은 독립이므로, Px^i 는 다음의 수식 (4)와 같이 정의하였다.

$$Px^i = 1 - (1 - Px_1^i)(1 - Px_2^i)\cdots(1 - Px_a^i) \tag{4}$$

만약 시간 구간 T_2 에서 영구적 결함이 발생하게 되면, 1 번째 반복 실행의 산술 연산들 중에서 x_1^1 만을 신뢰할 수 있다. 때문에 e 번의 반복 실행을 하는 동안

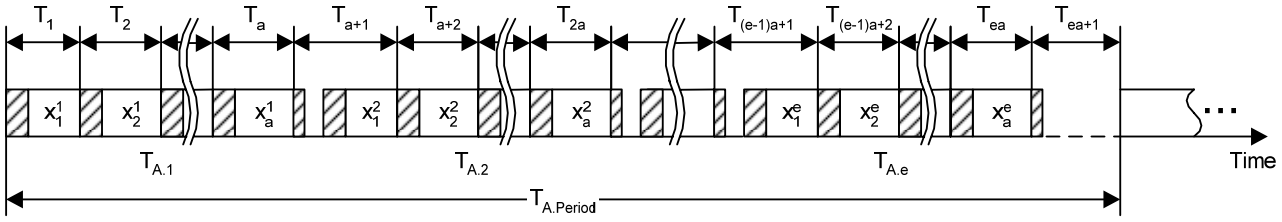

그림 2. E-TED 를 사용한 작업의 처리 과정

각 반복 실행의 1 번째 산술 연산들 (i.e., $x_1^2, x_1^3, \cdots, x_1^e$)의 결과값을 x_1^1 의 결과값과 비교하여 봄으로써 데이터 오류를 감지할 수 있으며, 감지 확률($P(D)$)은 다음의 수식 (5)와 같다.

$$P(D) = P_2 \cdot (1-(1-Px_1^2)(1-Px_1^3)\cdots(1-Px_1^e)) \quad (5)$$

마찬가지로 T_3 구간에서 결함이 발생하는 경우에는 x_1^1, x_2^1 만을 신뢰하며, 각 반복 실행의 1,2 번째 산술 연산들 중 어느 하나라도 결함이 발현하게 되면 데이터 오류를 감지할 수 있다. $T_2 \sim T_{a+1}$ 구간에서 결함이 발생한 경우에는 같은 방법을 이용하여 데이터 오류 감지 확률을 계산할 수 있지만, $T_{a+2} \sim T_{N-1}$ 구간의 경우에는 다음의 표 1 에서 보는 바와 같이 데이터 오류 감지 확률을 계산할 수 있다. 한편 T_1 구간에서 결함이 발생하는 경우에는 모든 산술 연산들을 신뢰할 수 없기 때문에 데이터 오류 감지 확률을 계산할 수 없으며, 마지막 구간 T_N 의 경우에는 반복 실행이 모두 종료된 이 후에 결함이 발생하였으므로 데이터 오류를 감지할 수 없다.

표 1. 시간 구간에 따른 데이터 오류 감지 확률

시간 구간	데이터 오류 감지 확률
T_1, T_N	데이터 오류 감지 불가
T_2	$P_2 \cdot (1-(1-Px)^{e-1})$
T_3	$P_3 \cdot (1-(1-Px)^{2(e-1)})$
…	…
T_{a+1}	$P_{a+1} \cdot (1-(1-Px)^{a(e-1)})$
T_{a+2}	$P_{a+2} \cdot (1-(1-Px)^{a(e-1)-1})$
T_{a+3}	$P_{a+3} \cdot (1-(1-Px)^{a(e-1)-2})$
…	…
T_{2a+1}	$P_{2a+1} \cdot (1-(1-Px)^{a(e-2)})$
…	…
T_{ea-1}	$P_{ea-1} \cdot (1-(1-Px)^2)$
T_{ea}	$P_{ea} \cdot Px$

표 1 에서 각 구간별 데이터 오류 감지 확률은 A2 와 수식 (3)에 의해 계산하였으며, 작업을 실행하는 모든 시간 구간, $T_{A.Period}$ 에 대해 데이터 오류를 감지할 확률은 각 시간 구간에서 데이터 오류를 감지할 확률을 모두 합한 것이며, 이는 다음의 수식 (6)과 같다.

$$P(D_x) = \left(\sum_{i=1}^{a} P_{i+1} \cdot \left[1-(1-Px)^{i(e-1)}\right] \right) + \left(\sum_{i=1}^{N-a-2} P_{i+a+1} \cdot \left[1-(1-Px)^{N-(i+a+1)}\right] \right) \quad (6)$$

이처럼 E-TED 를 이용하여 데이터 오류를 감지할 확률 식은 수식 (6)와 같이 표현되며, $e = 2$, $a = 3$ 인 경우 수식 (6)는 기존의 TED 를 이용하여 계산한 데이터 오류 감지 확률 식을 만족하는 것을 확인하였다.

5. 성능 평가

TED 와 E-TED 의 데이터 오류 감지 확률을 비교하는 실험을 위해 3 개의 산술 연산으로 이루어진 임의의 작업을 정의하였다. 또한 작업의 반복 실행에 대해 다음의 표 2 와 같이 각각의 시간 구간에서의 실행 시간을 정의하였다. 한편 임의의 산술 연산에서 결함이 발현할 확률은 수식 (3)에 의해 일정한 값(Px)으로 정의되며, 본 실험에서는 $10^{-3} \le Px \le 10^{-1}$ 으로 정의하였다.

표 2. 작업의 각 시간 구간에서의 실행 시간

시간 구간	실행 시간(ms)
T_1	3.7
$T_2, T_5, \cdots, T_{N-2}$	4.1
$T_3, T_6, \cdots, T_{N-1}$	3.4
$T_4, \cdots, T_{N-3}$	4.5
T_N	t_N

특히 마지막 시간 구간, T_N 에서의 실행 시간(t_N) 은 총 실행 시간에서 나머지 시간 구간($T_1 \sim T_{N-1}$)의 실행 시간들을 뺀 시간이며, 이는 다음의 수식 (7)과 같다.

$$t_N = T_{A.Period} - (4.1 + 3.4 + 4.5)e + 0.5 \\ = T_{A.Period} - 12e + 0.5 \quad (7)$$

수식 (2)에 의해, 각 시간 구간(T_i)에서 결함이 발생할 확률(P_i)은 각 실행 시간을 $T_{A.Period}$ 로 나누어 얻은 몫과 같다. 그림 4 는 $T_{A.Period}$ 가 증가함에 따른 E-TED 의 데이터 오류 감지 확률의 변화를 보여주고 있다($a = 3$, $10^{-3} \le Px \le 10^{-1}$). E-TED 는 T_d 가 일정 값(i.e., $3T_e$,$4T_e$)보다 증가하게 되면 반복 실행의 수도 함께 증가하게 되므로 데이터 오류 감지 확률도 단계적으로 증가하는 것을 확인하였다.

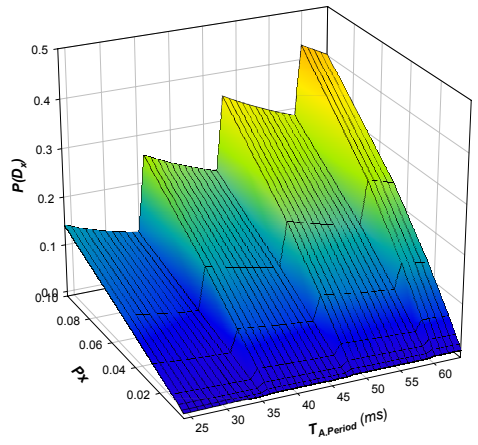

그림 4. $T_{A.Period}$ 의 증가에 따른 E-TED 의 데이터 오류 감지 확률

E-TED 를 이용하여 데이터 오류를 감지할 확률은 반복 실행의 수가 2 일 경우, TED 와 같은 감지 확률을 가진다(그림 5 참조). 그러나 반복 실행의 수가 3 이상일 경우(i.e., $T_d - T_s \ge 3T_e$), TED 는 반복 실행의 수가 항상 2 로 고정되어 있으므로 T_d 가 증가할수록 데이터 오류를 감지할 수 없는 시간 구간(예를 들어 그림 1 의 T_7)도 같이 증가하기 때문에 데이터 오류 감지 확률이 점점 낮아지게 된다.

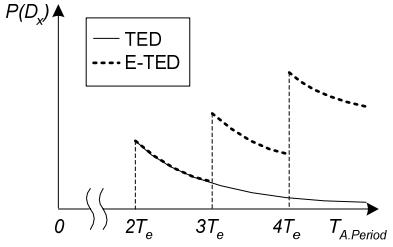

그림 5. TED 와 E-TED 의 데이터 오류 감지 확률

6. 결론

본 연구에서는 하드웨어적으로 여분을 두기 어려운 임베디드 시스템의 신인도를 향상시키기 위해, 소프트웨어 결함 허용 기법의 일종인 시간 여분 개념을 적용한 E-TED 기법을 연구하였으며, 이를 위해 E-TED 를 이용한 임베디드 시스템의 데이터 오류 감지 확률을 정의하였다. 임의의 작업에 대한 데이터 오류 감지 실험을 통해 E-TED 기법이 기존의 TED 기법에 비해 데이터 오류를 감지할 확률이 높은 것을 확인하였으며, 추후에는 작업의 실행 시간 동안 우선순위가 높은 다른 작업에 의해 인터럽트(Interrupt)가 발생하는 경우의 데이터 오류 감지 확률에 대한 연구를 수행할 예정이며, 또한 감지된 데이터 오류를 선택적으로 감춤(Masking) 또는 제거(Removal)하여 임베디드 시스템의 신인도를 향상 시킬 수 있는 메커니즘 개발에 대한 연구를 수행할 예정이다.

참고 문헌

[1] 박기진, 김광섭, 최석호, "고신뢰성 차량 임베디드 컴퓨팅 시스템의 백업 최소화 방안," 한국신뢰성학회 2005 학술발표대회 논문집, pp 295-301, June 2005.

[2] A. Avizienis, et al., "Fundamental Concepts of Dependability," *Research Report N01145*, LAAS-CNRS, Apr. 2001.

[3] Johnson B.W., *Design and Analysis of Fault-tolerant Digital Systems*, Addison-Wessley, 1989.

[4] Oh. N., and E. J. McCluskey, "Procedure Call Duplication: Minimization of Energy Consumption with Constrained Error Detection Latency" *Proc. IEEE Int'l Symp. on Defect and Fault Tolerance in VLSI Systems*, pp. 182-187, 2001.

[5] J. Aidemark, J. Vinter, P. Folkesson, and J. Karlsson, "Experimental Evaluation of Time-redundant Execution for a Brake-by-wire Application," *Int'l. Conf. on Dependable Systems and Networks*, Washington DC, USA, pp. 210-215, June 2002.

[6] J. Aidemark, P. Folkesson, and J. Karlsson, "On the Probability of Detecting Data Errors Generated by Permanent Faults Using Time Redundancy," *On-Line Testing Symposium*, pp. 68-74, July 2003.