총이온화선량에 의한 고장이 존재하는 비동기 순차 회로의 교정 제어

Corrective Control of Asynchronous Sequential Circuits with Faults from Total Ionizing Dose Effects in Space

양 정 민, 곽 성 우*

(Jung-Min Yang¹ and Seong Woo Kwak²)

¹Catholic University of Daegu

²Keimyung University

Abstract: This paper presents a control theoretic approach to realizing fault tolerance in asynchronous sequential circuits. The considered asynchronous circuit is assumed to work in space environment and is subject to faults caused by total ionizing dose (TID) effects. In our setting, TID effects cause permanent changes in state transition characteristics of the asynchronous circuit. Under a certain condition of reachability redundancy, it is possible to design a corrective controller so that the closed-loop system can maintain the normal behavior despite occurrences of TID faults. As a case study, the proposed control scheme is applied to an asynchronous arbiter implemented in FPGA.

Keywords: asynchronous sequential circuits, corrective control, fault diagnosis and tolerance, TID (Total Ionizing Dose)

I. 서론

현대 전자공학의 총아인 디지털 시스템은 아날로그 시스템에 비해서 많은 장점을 지닌다. 그 중 가장 뚜렷한 것은 디지털 시스템은 논리 회로(logic circuit)의 선천적 특성상 아날로그 회로보다 외란(disturbance)에 의한 고장에 매우 강하다는 점이다. 하지만 최근 디지털 시스템의 적용 환경이 심해(深海), 원자력 발전소, 우주 공간 등으로 점점 확대되면서 예측 못한 고장 요인이 새롭게 등장하였다. 예전에는 디지털 시스템에서 별로 고려되지 않았던 고장 탐지 및 극복(fault diagnosis and tolerance) 기술이 필수적으로 포함되어야 할 중요한 요소로 자리 잡았다.

이번 논문에서는 총이온화선량, 즉 TID (Total Ionizing Dose)의 영향으로 발생하는 고장이 존재하는 디지털 시스템에 대한 고장 극복 기법을 제안한다. TID 고장은 우주 방사선(cosmic radiation)이 시스템에 누적되어 일으키는 여러 가지 고장을 통칭하는 개념으로, 실리콘 반도체로 제작되어 우주 환경에서 작업하는 모든 디지털 시스템에서 일어날 수 있다[1,2].

본 논문에서는 순차 회로(sequential circuit)가 TID의 영향을 받아 회로의 상태 천이(state transition) 특성이 바뀌는 고장을 다룬다. CMOS 등 전자소자에 지속적으로 방사선이

가해지면 순차 회로를 구성하는 소자의 게이트 지연 시간(gate propagation delay)이 느려지며 이것은 결국 회로 상태 천이 특성의 일부가 바뀌는 결과를 낳는다. 순차 회로가 원래 이동해야 할 상태가 아닌 다른 상태로 천이하여 원래 상태로 즉시 복귀하지 않는다면 이후 입력에 대한 후속 동작은 계속 failure가 될 것이다.

이번 연구에서는 순차 회로 중에서도 전역 클럭(clock) 없이 동작하는 비동기 순차 회로(asynchronous sequential circuit)를 제어 대상 시스템으로 설정한다. 비동기 순차 회로는 동기(synchronous) 순차 회로에 비해 속도나 전력 사용 면에서 장점이 있기 때문에 디지털 시스템이나 통신 프로토콜(protocol) 등의 분야에서 여전히 널리 사용되고 있다[3]. 또 우주용 디지털 시스템에서도 카운터(counter), 오류 검출 및 정정(EDAC: Error Detection and Correction) 회로의 형태로 활용되고 있다[4].

TID 고장을 극복하는 방법으로서 비동기 순차 회로를 위한 교정 제어(corrective control) 기법을 사용한다. 교정 제어는 전통적인 피드백 제어(feedback control)를 이용하여 비동기 순차 회로의 폐루프(closed-loop) 시스템 동작을 보상한다. 비동기 회로 앞에 설치된 제어기는 외부 입력과 출력 피드백을 받아서 제어 입력을 생성하여 비동기 회로로 보냄으로써 머신이 나타내는 올바른지 않은 동작을 교정한다. 교정 제어는 이미 설계된 비동기 순차 회로의 안정 상태 동작(stable-state behavior)을 바꿀 수 있다는 점에서 학계의 주목을 받고 있다. 교정 제어를 이용하여 여러 가지 결함이 존재하는 비동기 회로의 동작을 보정하는 선행 연구가 Hammer 등[5-7]과 저자의 논문[8-10]에서 발표되었다.

* 책임저자(Corresponding Author)

논문접수: 2011. 8. 20., 수정: 2011. 9. 5., 채택확정: 2011. 9. 25.

양정민: 대구가톨릭대학교 전자공학과(jmyang@cu.ac.kr)

곽성우: 계명대학교 전자공학과(ksw@kmu.ac.kr)

※ 이 논문은 2011년도 정부(교육과학기술부)의 재원으로 한국연구재단의 기초연구사업 지원을 받아 수행된 것임(No. 2011-0005116).

본 논문에서는 먼저 TID 영향에 의한 상태 천이 고장을 비동기 순차 회로 내에서 모델링하고 교정 제어에 의한 고장 극복이 가능한 필요충분조건을 해석적으로 규명한다. TID 고장에 관계없이 정상적인 동작을 유지하기 위해서는 비동기 순차 회로 내부에서 여분의 도달가능성(redundant reachability)이 존재해야 한다. 그런 다음 상태 피드백 교정 제어기를 설계하여 고장 극복 과정을 제시한다. 또 사례 연구로서 우주용 디지털 시스템에서 많이 사용되는 arbiter를 비동기적으로 설계하고 고장 시나리오를 실험적으로 만들어 제안된 기법의 실용 가능성을 검증한다.

비동기 머신의 교정 제어에 대한 기존 연구에서 크리티컬 레이스(critical race) [5,7], 무한 순환(infinite cycle) [6], 입력 외란[8,10] 등의 고장 요인이 다루어졌으나 우주용 디지털 시스템에서 발생하는 TID 영향에 의한 고장은 이번 연구에서 최초로 다루는 주제이므로 본 연구는 선행 연구와 차별성을 가진다. 또한 사례 연구에서 FPGA 실험을 수행함으로써 교정 제어가 실제 비동기 순차 회로를 위한 고장 극복 기법의 하나로 쓰일 수 있다는 사실을 보인 점도 본 논문의 기여도라 말할 수 있다.

II. TID 고장이 존재하는 비동기 순차 회로

본 논문에서는 회로의 출력이 현재 상태와 동일한 입력/상태 비동기 순차 회로에 대해서 다루기로 한다. 입력/상태 비동기 회로 Σ 를 유한 상태 머신(finite state machine)으로 표현하면 다음과 같다.

$$\Sigma = (A, X, x_0, f)$$

위 식에서 A 는 입력 집합, X 는 상태 집합, $x_0 \in X$ 은 초기 상태이며, $f: X \times A \rightarrow X$ 는 상태 천이 함수이다. f 가 정의되는 상태-입력 조합 $(x, v) \in X \times A$ 가 $f(x, v) = x$ 이면 안정 조합(stable combination), $f(x, v) \neq x$ 이면 과도 조합(transient combination)이라고 정의한다. 이때 상태 x 는 각각 ‘안정 상태’와 ‘과도 상태’로 불린다. 비동기 회로 Σ 가 안정 조합 (x, v) 에 있으면 입력이 바뀌지 않는 한 계속 안정 상태 x 에 머문다. 입력이 v' 에서 v 로 바뀌고 (x, v) 가 과도 조합이라고 하면 Σ 는 $x_1 := f(x, v)$, $x_2 := f(x_1, v)$, ... 등으로 여러 개의 과도 상태를 거치며 $x' := f(x', v)$ 를 이루는 ‘다음 안정 상태(next stable state)’ x' 로 천이한다[11].

클럭 없는 비동기 회로 Σ 가 (x, v) 에서 (x', v) 까지 천이하는 시간은 순간적이며 이론적으로는 0이다. 따라서 외부에는 Σ 가 중간에 거치는 과도 상태 $x_1, x_2, \dots$ 가 관측되지 않는다. 이 성질을 표현하기 위해서 ‘stable recursion 함수’ s 를 다음과 같이 정의한다[5].

$$s: X \times A \rightarrow X, s(x, v) = x', x, x' \in X, v \in A$$

$s(x, v)$ 는 (x, v) 에 있던 Σ 가 이동하여 도달하는 다음 안정 상태 x' 를 출력한다. f 대신 s 를 이용하여 단위 입력 대신 입력 스트링(string)을 s 의 변수로 설정하면 다음과 같이 일반화할 수 있다.

$$s(x, vt) = s(s(x, v), t), x \in X, v \in A, t \in A^+$$

위 식에서 A^+ 는 A 에 속한 알파벳으로 이루어지는 길이

1 이상의 스트링 집합을 말한다. 위 정의를 이용하여 ‘안정적 도달가능성(stable reachability)’ 성질[5,7]을 아래와 같이 정의한다. 이 성질은 뒤에서 고장 극복 교정 제어의 존재조건 규명 과정에서 사용된다.

정의 1: $s(x, t) = x'$ 인 입력 스트링 $t \in A^+$ 가 존재하면 상태 x' 는 상태 x 로부터 ‘안정적으로 도달 가능’하다.

서론에서 기술했듯이 TID 영향에 의한 고장은 회로의 상태 천이 특정 일부가 영구적으로 바뀌는 현상을 말한다. TID 고장을 나타내기 위하여 비동기 회로 Σ 에서 TID 고장이 발생하는 과도 조합이 $(x_1, v_1), \dots, (x_r, v_r)$ 로 r 개 있다고 가정한다. 또 (x_i, v_i) 에서 $(1 \leq i \leq r)$ TID 고장이 발생하여 Σ 가 다음 안정 상태 $s(x_i, v_i)$ 대신 이동하는 올바른지 않은 상태를 z_i 라 하자. Σ 의 TID 고장 명세는 아래와 같은 집합 $F \subset X \times A \times X$ 로 표현된다.

$$F := \{(x_1, v_1, z_1), (x_2, v_2, z_2), \dots, (x_r, v_r, z_r)\}$$

F 에 속한 상태 x_i 가 이루는 어떤 안정 조합 (x_i, u) 에 Σ 가 머물러 있다고 가정하고 TID 고장의 발생 과정을 설명한다. 외부 입력이 u 에서 v_i 로 바뀐다고 하자. 만약 TID 고장이 아직 발생하지 않았다면 Σ 는 과도 상태를 거쳐서 다음 안정 상태 $s(x_i, v_i)$ 로 도달하는 정상적인 상태 천이를 할 것이다. 그런데 방사능 누적이 지속되어 임계점을 지나 시스템 특성이 (x_i, v_i) 에서 바뀌었다고 하자. 그렇다면 입력 v_i 를 받은 Σ 는 $s(x_i, v_i)$ 대신 잘못된 상태 z_i 로 이동하여 (고장 극복이 선행되지 않으면) failure가 된다.

위에서 설정한 F 값, 즉 비동기 순차 회로에서 TID 영향에 의한 고장이 발생하는 상태-입력 조합과 고장의 결과는 미리 알 수 있다[1,2]. 문제는 TID 고장이 언제 일어나는지 그 발생 시점은 알 수 없다는 점이다. 우주 방사능 누적량은 모델링이 거의 불가능하여 정확한 예측 값을 알지 못하기 때문이다. 따라서 TID 고장이 비동기 회로 구동 시작 후 언제든지 일어날 수 있다는 가정 하에 고장 극복 제어기를 설계해야 한다.

III. 고장 극복 교정 제어

1. 페루프 시스템

그림 1은 입력/상태 비동기 순차 회로에 대한 교정 제어 페루프 시스템의 기본 구조이다. C 는 교정 제어기이며 Σ 와 마찬가지로 비동기 회로로 구현된다. $v \in A$ 는 외부 입력, $u \in A$ 는 C 가 만드는 제어 입력이며, $x \in X$ 는 Σ 로부터 나오는 상태 피드백 값이다. Σ_c 는 제어기 C 와 비동기 카운터 Σ 로 구성된 페루프 시스템을 가리킨다. 그림 1의 교정 제어기 C 는 출력 값이 현재 상태 값과 다르므로 다음과 같은 입력/출력 비동기 머신으로 표현된다.

$$C = (X \times A, A, \Xi, \xi_0, \phi, \eta)$$

$X \times A$ 는 C 의 입력 집합, A 는 C 의 출력 집합을 가리킨다. 또 Ξ 는 C 의 상태 집합, $\xi_0 \in \Xi$ 는 초기 상태이며, $\phi: \Xi \times X \times A \rightarrow \Xi$ 와 $\eta: \Xi \times X \times A \rightarrow A$ 는 각각 C 의 상태 천이 함수와 출력 함수이다.

그림 1에서 “TID fault”는 TID 영향에 의한 고장을 가리

그림 1. TID 영향에 의한 고장을 극복하기 위한 교정 제어 시스템.

Fig. 1. Corrective control system for tolerating faults caused by TID effects.

킨다. 제어기 C가 TID 고장을 탐지하고 복구하는 과정은 다음과 같다. 고장이 일어나기 전에는 C가 외부 입력 v 를 그대로 제어 입력 u 로 전달하여 Σ 에 보낸다. Σ 가 정상 동작을 한다면 상태 피드백 값 x 도 역시 올바른 값으로 나온다. x 를 받은 C는 외부 입력 v 가 바뀌면 계속 $u=v$ 로 할당한다. 다시 말하면 고장이 일어나기 전에는 C가 아무런 실제적인 제어 과정을 실행하지 않는다.

TID 고장이 $(x_i, v_i, z_i) \in F$ 인 상태-입력 조합 (x_i, v_i) 에 발생했다고 가정하고 Σ 가 안정 상태 x_i 에 있을 때 외부 입력이 v_i 로 바뀌었다고 하자. C는 아직 고장 발생을 감지하지 못하므로 제어 입력을 $u=v_i$ 로 해서 Σ 에 보낸다. 하지만 TID 고장으로 인해서 다음 안정 상태는 $s(x_i, v_i)$ 가 아닌 z_i 가 되며 이 값이 상태 피드백 $x=z_i$ 로 나와 제어기 C에 들어간다. z_i 를 관측해서 고장을 탐지한 C는 외부 입력 v 가 다시 바뀌기 전에 Σ 가 z_i 에서 올바른 상태 $s(x_i, v_i)$ 로 가도록 하는 교정 제어 동작을 실시한다. 그림 1의 폐루프 시스템은 클럭 없이 움직이는 비동기 머신이므로 C가 고장을 탐지하고 고장 극복 교정 제어를 완료하기까지의 과정은 모두 비동기적으로 순간적으로 실행된다[5-7]. 따라서 외부 사용자에게는 Σ 가 잘못된 상태 z_i 로 천이되는 현상이 보이지 않고 폐루프 시스템 Σ_c 가 정상 동작을 유지하는 것처럼 관측된다.

2. 교정 제어기: 존재조건 및 설계 과정

고장 극복 교정 제어기가 존재할 필요충분조건을 규명한다. Σ 가 과도 조합 (x_i, v_i) 에서 일어나는 TID 고장을 극복하기 위해서는 상태 z_i 에서 상태 $s(x_i, v_i)$ 까지 비동기적으로 이동해야 한다고 하였다. 교정 제어에서 이 과정을 실현하기 위해서는 $s(x_i, v_i)$ 가 z_i 로부터 안정적으로 도달 가능해야 한다 [5]. 즉 $s(z_i, t_i)=s(x_i, v_i)$ 인 입력 스트링 $t_i \in A^+$ 가 존재해야 한다. 또 F에 속한 임의의 상태-입력 조합 (x_i, v_i) 에 대해서 이 조건이 모두 성립되어야 한다. 고장 극복 교정 제어기 C의 존재조건을 표현하면 아래와 같다.

정리 1: TID 영향으로 일어나는 상태 천이 고장 F가 존재하는 입력/상태 비동기 순차 회로 $\Sigma=(A, X, x_0, f)$ 의 폐루프 시스템 Σ_c 가 그림 1과 같을 때 다음 두 명제 (a)와 (b)는 서로 동가(等價)이다.

(a) Σ_c 가 TID 고장에 대해서 내고장성을 가지게 하는 교정 제어기 C가 존재한다.

(b) $\forall i=1, \dots, r, \exists t_i$ such that $s(z_i, t_i)=s(x_i, v_i)$.

고장 극복 교정 제어기의 설계 과정은 크리티컬 레이스를 없애는 문제[5]나 제어 불능 입력(uncontrollable input)이 존재할 때의 모델 정합(model matching) 문제[8]와 유사하다. F에 속한 임의의 상태-입력 조합 (x_i, v_i) 를 예로 들어 제어기 설계 과정을 기술한다. 다른 상태-입력 조합에 대한 제어기 모듈도 유사하게 설계된다. F에 속한 각 상태-입력 조합에 대한 제어기 모듈을 모두 설계한 후 [6,9] 등에서 제시된 방법을 이용하여 하나로 통합하면 완전한 C가 완성된다.

초기 상태 ξ_0 에 있던 제어기 C는 Σ 가 안정 상태 x_i 에 진입하는 순간 미리 정의된 상태 $\xi_i \in E$ 로 이동하여 TID 고장 발생에 대비한다. 앞에서 기술했듯이 제어기 C가 TID 고장의 발생 시점을 미리 알 수 없기 때문에 C는 TID 고장이 항상 일어났다고 가정하고 제어 과정에 임할 수밖에 없다. 이때 외부 입력 v 가 v_i 가 아닌 값으로 바뀐다면 C는 초기 상태 ξ_0 로 복귀한다. 반대로 v 가 TID 고장이 발생했는지 모르는 상태-입력 조합을 만드는 v_i 로 바뀌었다고 하자. C는 일단 v_i 를 제어 입력 u 로 그대로 전달하고 Σ 의 동작 결과를 기다린다. 만약 그림 1에서 C가 상태 피드백 값 $x=s(x_i, v_i)$ 를 받았다면 이것은 TID 고장이 아직 일어나지 않았다는 사실을 의미한다. 따라서 C는 다시 ξ_0 로 복귀하며 Σ 는 정상 동작을 지속한다.

입력 v_i 가 들어온 후 상태 피드백 값이 $s(x_i, v_i)$ 가 아니라 z_i 로 나왔다고 하자. C는 z_i 를 받은 순간 TID 고장의 발생을 감지하고 즉시 고장 극복 교정 동작을 시작한다. 정리 1에서 교정 제어기 C를 설계할 수 있는 필요충분조건이 $s(z_i, t_i)=s(x_i, v_i)$ 인 입력 스트링 t_i 의 존재라고 밝혔다. t_i 를 이용하여 C의 교정 동작을 구현한다.

$t_i:=u_1 u_2 \dots u_m$ 이라고 하자. Σ 가 안정 상태 z_i 에서 입력 $u_1, u_2, \dots, u_m$ 을 차례로 받아서 거치는 중간 상태들을 $y_1, y_2, \dots, y_{m-1} \in X$ 이라고 하면 아래 관계가 성립한다.

$$\begin{aligned} s(z_i, u_1) &= y_1 \\ s(y_1, u_2) &= y_2 \\ &\vdots \\ s(y_{m-2}, u_{m-1}) &= y_{m-1} \\ s(y_{m-1}, u_m) &= s(x_i, v_i) \end{aligned}$$

t_i 에 대응하여 C도 m 개의 보조 상태 $\xi_1, \xi_2, \dots, \xi_m \in E$ 를 정의한다. C의 교정 동작은 다음과 같다. 상태 피드백 z_i 를 받은 순간 C는 첫 번째 보조 상태 ξ_1 로 천이한다. ξ_1 에서 C는 t_i 의 첫 번째 원소 u_1 을 제어 입력으로 Σ 에 전달한다. 위 식에 따라서 Σ 는 z_i 에서 첫 번째 중간 상태 y_1 로 이동하며 상태 피드백 값도 y_1 로 바뀐다. $x=y_1$ 로 되는 순간 C는 두 번째 보조 상태 ξ_2 로 천이한 후 t_i 의 두 번째 원소 u_2 를 입력시킨다. u_2 를 받은 Σ 도 역시 두 번째 중간 상태 $s(y_1, u_2)=y_2$ 로 천이한다. t_i 의 마지막 원소 u_m 까지 이 동작이 연속적으로 실행되면 Σ 는 목적 상태 $s(x_i, v_i)$ 에 도달하게 된다.

이러한 일련의 교정 제어 과정은 모두 클럭 없이 비동기적으로 이루어지므로 실행 속도는 매우 빠르다. 따라서 폐루프 시스템 Σ_c 는 안정 상태 x_i 에서 외부 입력 v_i 를 받고 즉시 정상적인 다음 안정 상태 $s(x_i, v_i)$ 로 천이하는 것처럼 관측되며 고장 극복 과정이 완료된다. C의 상태 천이 함수

그림 2. 고장 극복 교정 제어 과정.

Fig. 2. Corrective control procedure for fault tolerance.

ϕ 와 출력 함수 η 를 이용하여 앞에서 기술한 제어 동작을 표현하면 아래와 같다.

$$\begin{aligned} \phi(\xi_0, Z_i, V_i) &= \xi_1 \text{ (상태 피드백 } Z_i \text{가 들어오면 } \xi_1 \text{로 이동)} \\ \eta(\xi_1, Z_i, V_i) &= u_1 \text{ (출력으로 } u_1 \text{ 생성)} \\ \phi(\xi_1, Y_1, V_i) &= \xi_2 \text{ (}\Sigma \text{가 } Y_1 \text{로 이동하면 } \xi_2 \text{로 이동)} \\ \eta(\xi_2, Y_1, V_i) &= u_2 \text{ (출력으로 } u_2 \text{ 생성)} \\ &\vdots \\ \eta(\xi_m, Y_{m-1}, V_i) &= u_m \text{ (}\xi_m \text{에서 출력으로 } u_m \text{ 생성)} \\ \phi(\xi_m, S(X_i, V_i), V_i) &= \xi_0 \\ (\Sigma \text{가 목적 상태 } S(X_i, V_i) \text{에 도달하면 } C \text{는 } \xi_0 \text{로 복귀}) \end{aligned}$$

TID 고장이 한 번 발생한 후에는 상태 천이 특성이 영구적으로 바뀌므로 제어기는 Σ 가 (X_i, V_i) 에 진입할 때마다 항상 정의한 교정 동작을 항상 실행한다. 이 개념을 도식적으로 나타내면 위 그림 2와 같다.

IV. 사례 연구: 비동기 Arbiter

1. 비동기 Arbiter 구조

위성의 데이터 저장 장치(Solid State Data Recorder: SSDR)는 탑재체(payload)로부터 데이터를 수집하여 지상으로 전송하기 전까지 데이터를 임시로 저장하는 역할을 한다. 일반적으로 인공위성은 다수의 탑재체를 가진다. 위성 데이터 저장 장치는 각 탑재체로부터 전송되는 데이터를 받아 정해진 메모리 영역에 저장한다. 이때 특정 시각에 하나의 탑재체만 메모리에 접근할 수 있으므로 다수의 탑재체로부터 데이터를 수신하여 저장하기 위해서는 메모리 접근을 통제할 arbiter 모듈이 필요하다. Arbiter 모듈은 탑재체로부터 메모리 접근 요구에 대응하여 각 탑재체가 메모리에 접근할 순서를 지정해준다. 최근 메모리 접근 속도를 높이기 위해서 전역 클럭을 쓰지 않고 탑재체의 요청 신호에

그림 3. 3개의 탑재체를 위한 데이터 저장 장치용 비동기 arbiter의 상태 흐름도.

Fig. 3. State flow diagram of the asynchronous SSDR arbiter for three payloads.

따라서 즉각 반응하는 비동기적 arbiter 모듈도 많이 등장하고 있다[12,13].

그림 3은 위성에 3개의 탑재체 A, B, C가 있는 경우 대한 비동기 arbiter의 상태 흐름도(state flow diagram)를 나타낸 것이다. 상태 S_a 는 탑재체 A가 메모리에 접근하도록 하며, 상태 S_b 와 S_c 는 각각 탑재체 B와 C가 메모리에 접근하도록 한다. 상태 S_i 는 어떠한 탑재체도 메모리에 접근하지 않는 유휴(idle) 상태를 나타낸다. 여러 메모리 접근 요구가 동시에 있을 경우 이를 통제하는 방식으로는 우선순위(priority) 할당 방식과 균등 시간(round robin) 할당 방식이 있다. 모든 탑재체들에 동일한 기회를 주기 위해서 일반적으로 균등 시간 할당 방식을 많이 사용한다[14]. 예를 들어 탑재체 A, B, C가 동시에 메모리 접근 요구를 하는 경우, arbiter는 먼저 S_a 로 천이하여 탑재체 A가 메모리에 접근하도록 허락한다. 탑재체 A에 할당된 시간이 지난 후 arbiter는 A의 접근을 중지시키고 탑재체 B가 메모리에 접근할 수 있도록 S_b 로 천이하며, 그 다음에는 S_c 로 천이하여 탑재체 C가 접근하도록 한다. 이후 다시 $S_a \rightarrow S_b \rightarrow S_c$ 순서로 상태 천이를 하여 모든 탑재체가 동등한 메모리 접근 기회를 갖도록 한다. arbiter 안의 어떤 상태에서 다른 상태로의 천이는 그림 3에서 표시된 입력 신호 “ v_{jk} ”에 의해 결정된다($j, k \in \{a, b, c, i\}$). 예를 들어 v_{ab} 는 S_a 에서 S_b 로 천이시키는 입력 이벤트(event) 신호이며, v_{bc} 는 S_b 에서 S_c 로 천이시키는 이벤트 신호이다. 다른 신호들도 비슷하게 해석된다. 입력 이벤트 신호는 v_{ab} 에 대한 아래 식과 같이 탑재체가 arbiter에 주는 요청 신호들의 논리 조합으로 구성된다.

$$v_{ab} = \text{ReqA} \cdot \text{ReqB} + \text{ReqA} \cdot \text{ReqB} \cdot \text{TimeUp}$$

여기서 ReqA , ReqB 는 각각 탑재체 A, B의 메모리 요구를 나타내는 논리 신호이고, ReqA 는 탑재체 A의 요구가 없을 나타내는 논리기호이다. TimeUp 신호는 특정 탑재체에

그림 4. 플립플롭으로 구현한 비동기 arbiter의 상태 비트

Fig. 4. State bit of the asynchronous arbiter implemented in a flip-flop.

할당된 메모리 접근 시간이 지났음을 나타낸다. TimeUp 신호는 arbiter가 S_a , S_b , 또는 S_c 상태로 진입할 때 타이머를 동작시킨 후 경과 시간을 측정하여 발생 시킨다.

4개의 상태를 가진 그림 3의 유한 상태 머신의 상태를 디지털 회로로 구성하기 위해서는 두 개의 상태 비트(bit)가 필요하며, 이 비트들은 플립플롭(flip-flop) 또는 래치(latch)로 구현된다. 각 플립플롭의 입력(D)에는 그림 4와 같이 조합 회로(combinational circuit)의 논리 연산 결과가 들어간다. 조합 회로는 외부 입력 신호와 플립플롭의 출력 신호를 기반으로 다음 상태 천이를 위한 신호를 생성한다. 하지만 조합 회로에 방사능이 누적되어 TID 효과가 발생하면 회로의 출력 신호가 지연된다. 출력 지연으로 인해 플립플롭에 올바른 값이 입력되지 않으면 원하는 상태 천이가 불가능하게 하게 된다. 그림 4에서는 TID 영향을 받은 AND 게이트의 출력이 1에서 0으로 변하는 시점이 점선으로 표시된 것처럼 늦어져 플립플롭에 저장되어야 할 값이 “0”으로 바뀌지 않고 계속 “1”로 유지되는 현상을 보여준다. 이와 같이 시간 지연에 의해 발생할 수 있는 오류는 플립플롭에 저장된 값이 바뀌지 않고 현재 값으로 계속 머무르는 결과로 나온다.

이 같은 TID에 의한 상태 천이 고장을 극복하기 위해서는 그림 4에서 점선으로 표시한 부분과 같은 별도의 부가적인 제어 로직 및 제어 알고리즘이 필요하다. 다시 말하면 회로 입력에 대한 여유도(redundancy)가 요구된다. 본 연구에서는 TID 고장을 감내할 수 있는 입력 여유도를 가진 비동기 arbiter를 설계하고 앞서 제안한 비동기 교정 제어 기법을 적용하여 비동기 arbiter가 고장을 극복하는 과정을 보인다.

2. 내고장성 비동기 Arbiter 설계

TID 영향에 의한 상태 천이 고장이 존재하며 또한 고장 극복을 위한 입력 여유도를 가진 비동기 arbiter를 $\Sigma = (A, X, x_0, f)$ 라 하자. 입력 집합 A는 다음과 같이 정의된다.

$$A := \{v_{jk} \mid j, k \in \{a, b, c, i\}\} \cup \{c_{ab}, c_{ba}\}$$

v_{jk} 는 그림 3에서 설정한 arbiter 입력 신호들이다. c_{ab} 와 c_{ba} 는 고장 극복을 위해서 더해진 제어 로직 신호로서 아래와 같이 정의된다.

$$s(S_a, c_{ab}) = S_b$$

$$s(S_b, c_{ba}) = S_a$$

다시 말하면 Σ 는 그림 3의 흐름도에서 위의 두 상태 천이를 더한 유한 상태 머신으로 구성된다. Σ 가 가진 4개의 상태는 각각 다음과 같이 두 개의 비트로 표시한다.

$$X := \{S_i=00, S_a=10, S_b=01, S_c=11\}, x_0 := S_i$$

본 논문에서는 TID 영향에 의한 고장이 왼쪽 상태 비트에 발생한 경우에서 비동기 arbiter Σ 가 보이는 동작을 기술한다. 오른쪽 상태 비트에 대한 고장도 유사하게 설명된다. 상태 비트에 TID 고장이 발생하면 그림 4에 보인 것처럼 비트의 로직 값을 반전시키는 정상 입력 신호가 작동하지 않는다. 이 비트 값이 0이나 1이었을 때 TID 고장이 발생하면 어떤 입력 v_{jk} 에 대해서는 1이나 0으로 바뀌지 않고 이전 값을 그대로 유지한다. 즉 그림 3에서 나타난 정상적인 상태 천이가 이루어지지 않는 경우가 발생한다.

그림 5는 이러한 동작을 도시한 것이다. 예를 들어 안정 상태 $S_a=10$ 에서 왼쪽 상태 비트에 TID 영향에 의한 고장이 일어났다고 가정하자. 이때 외부 입력이 v_{ai} 로 바뀌었다고 하자. 정상적인 동작이라면 Σ 는 $s(S_a, v_{ai})=S_b$, 즉 다음 안정 상태 S_b 로 천이해야 한다. 하지만 TID 고장이 왼쪽 상태 비트에 일어났기 때문에 두 상태 비트는 $S_a=10$ 에서 $S_b=01$ 로 가지 못하고 대신 $S_c=11$ (왼쪽 비트가 1로 고정)로 바뀐다. 또 TID 고장 후 안정 상태 S_c 에서 외부 입력이 이번에는 v_{ai} 로 바뀌었다고 가정하자. 앞서와 마찬가지로 arbiter Σ 는 $S_a=00$ 으로 가지 못하고 상태 비트 10, 즉 S_a 에 그대로 머무르게 된다. 그림 5에 표시된 오류가 있는 상태 천이는 모두 이와 같이 해석될 수 있다. 그림 5로부터 II장에서 정의한 TID 고장 집합 F를 구하면 아래와 같다.

$$F = \{(S_a, v_{ab}, S_c), (S_a, v_{ai}, S_a), (S_b, v_{ba}, S_i), (S_b, v_{bc}, S_b), (S_c, v_{cb}, S_c), (S_c, v_{ci}, S_a), (S_i, v_{ia}, S_i), (S_i, v_{ib}, S_b)\}$$

다음으로 우리가 추가한 여유 입력 신호 c_{ab} 와 c_{ba} 가 있으면 F의 상태 천이 고장을 모두 극복할 수 있는 그림 1의 교정 제어기 C가 존재함을 보인다. 위에서 예로 든 상태 천이 고장 (S_a, v_{ab}, S_c) 를 계속 사용하여 설명한다. 정리 1에서

그림 5. 왼쪽 상태 비트에 TID 고장이 발생한 경우의 비동기 arbiter Σ 동작.

Fig. 5. Behavior of the asynchronous arbiter Σ after an occurrence of TID fault at the left-side state bit.

밝힌 바와 같이 페루프 시스템 Σ 가 TID 고장에 대해서 내 고장성을 가지게 하는 교정 제어기 C가 존재하기 위한 필요충분조건은 정상적인 다음 안정 상태 $s(S_a, V_{ab})=S_b$ 가 고장 때문에 천이하는 상태 S_c 로부터 안정적으로 도달 가능해야 한다는 것이다. 그런데 그림 5를 보면 $s(S_b, t)=S_c$ 인 입력 스트링 $t=v_{ca}c_{ab}$ 가 존재함을 알 수 있다. 만약 c_{ab} 가 없으면 이 조건은 성립되지 않는다. 따라서 여분의 입력 c_{ab} (또는 c_{ba})는 TID 고장 후에도 Σ 가 정상 동작을 하게 하는 도달 가능성을 유지시키는 역할을 한다. F에 속한 다른 상태 천이 조합에 대해서도 유사한 방법으로 입력 스트링을 찾는다. 따라서 정리 1에 의해서 비동기 arbiter Σ 를 위한 내고장성 교정 제어기 C를 만들 수 있다.

V. 실험 결과

앞 절에서 제안한 내고장성 비동기 arbiter와 상태 천이 고장을 극복하는 교정 제어기를 결합한 페루프 시스템을 VHDL로 프로그래밍 하고 FPGA 상에서 직접 구현하였다. 컴파일러로 QUARTUS II(ver. 9.1)를 사용하였으며 Altera社의 Cyclone FPGA를 탑재한 X-Hyper270 kit에 페루프 시스템을 porting하였다. 또 주요 외부 입력과 제어 입력, 그리고 상태 비트들의 동작 변화를 4-채널 디지털 오실로스코프를 통해서 추출한 후 시간에 대한 그래프로 나타내었다.

그림 6은 페루프 시스템이 상태 천이 고장 (S_a, V_{ab}, S_c)를 극복하는 과정을 실험으로 보인 결과이다. y_1, y_0 은 두 개의 상태 비트를 가리킨다. 시간 구간 $t = 0 \sim 120(\text{nsec})$ 에서 비동기 arbiter Σ 는 $y_1, y_0=10$, 즉 상태 S_a 에 머물러 있다. 그런데 이 사이 우리가 알지 못한 어떤 순간에 TID 영향에 의한 고장이 왼쪽 상태 비트 y_1 에 발생하였다. 시각 t_1 (약 130 nsec)에서 외부 입력 v_{ab} 가 들어왔지만 그림 6에서 볼 수 있듯이 Σ 는 올바른 다음 상태 $S_b=01$ 이 아니라 잘못된 상태 $S_c=11$ 로 천이한다. 이때 교정 제어기 C가 고장 발생을 감지하고 교정 동작을 시작한다. 앞 장에서 설계한 대로 C는 제어 입력 스트링 $t=v_{ca}c_{ab}$ 를 이용한다. 먼저 시각 t_2 에서 C는 t 의 첫 번째 입력 v_{ca} 를 생성하며, v_{ca} 를 받은 Σ 는 상태 $S_c=10$ 로 천이한다(그림 6의 구간 $[t_2, t_3]$ 참조). Σ 가 S_a 로 간 사실을 확인한 C는 다음 제어 입력 c_{ab} 를 생성하고, Σ 가 상태 $S_c=01$ 로 천이함으로써 제어 동작이 완성된다. 이 과정은 그림 6의 구간 $[t_3, t_4]$ 와 제어 입력 c_{ab} 의 변화를 통해 확인할 수 있다.

상태 천이 고장을 감지한 시각 t_1 에서부터 교정 제어 동작이 완료되는 시각 t_4 까지의 시간은 그림 6에서 약 30 nsec로 측정된다. 이 값은 일반 동기 시스템의 전역 클럭 주기보다 훨씬 작으므로 제안된 페루프 시스템이 비동기적으로 동작했음이 입증된다.

그림 7은 상태 천이 고장 (S_a, V_{ai}, S_a)에 대한 FPGA 실험 결과를 나타낸 것이다. 그림 5에서 $s(S_a, c_{ab}v_{bi})=S_i$ 가 성립되므로 교정 제어기 C는 $c_{ab}v_{bi}$ 를 제어 입력 스트링으로 사용하였고, 그림 7에서는 외부 입력 v_{ai} 와 제어 입력 v_{bi} 의 변화를 각각 나타내었다(c_{ab} 는 표시 안 함). 이 고장에 대한 교정 동작 및 그림 7에 표시된 Σ 의 상태 천이 변화 과정은 그림 6의 경우와 유사하게 해석할 수 있다.

그림 6. 상태 천이 고장 (S_a, V_{ab}, S_c)에 대한 FPGA 실험 결과.

Fig. 6. Result of FPGA experiment for the state transition fault (S_a, V_{ab}, S_c).

그림 7. 상태 천이 고장 (S_a, V_{ai}, S_a)에 대한 FPGA 실험 결과.

Fig. 7. Result of FPGA experiment for the state transition fault (S_a, V_{ai}, S_a).

VI. 결론

이번 연구에서는 TID 영향에 의한 상태 천이 고장이 존재하는 비동기 순차 회로의 고장 극복 기법을 제안하였다. 본 논문에서는 TID 고장이 발생하는 회로의 지점과 고장의 결과는 미리 예측 가능하나 고장이 발생하는 시점은 알지 못하는 일반적인 상황을 설정하였다. 비동기 회로가 TID 고장으로 인해서 상태 천이 특성에 영구적인 손상을 입으면 고장의 결과로 이동하는 상태로부터 원래 도달해야 하는 상태까지 여유 도달 가능성이 존재해야 고장 극복이 가능하다. 고장 극복 교정 제어기를 구현할 수 있는 필요충분조건을 규명하고 제어기 설계 과정을 제시하였다. 또한 비동기 arbiter라는 실제적인 디지털 회로를 사례 연구로 들어 본 논문에서 제안된 기법의 효용성을 검증하였다.

본 논문이 점차 수요가 늘고 있는 우주용 디지털 시스템의 고장 탐지 및 극복에 대한 연구에 기여하기를 기대한다.

참고문헌

- [1] H. L. Hughes and J. M. Benedetto, "Radiation effects and hardening of MOS technology: devices and circuits," *IEEE Transactions on Nuclear Science*, vol. 50, no. 3, pp. 500-521, Jun. 2003.
- [2] E. O. Mikkola, B. Vermeire, H. G. Parks, and R. Graves, "VHDL-AMS modeling of total ionizing dose radiation effects on CMOS mixed signal circuits," *IEEE Transactions on Nuclear Science*, vol. 54, no. 4, pp. 929-934, Aug. 2007.
- [3] J. Sparsø and S. Furber, *Principles of Asynchronous Circuit design - A Systems Perspective*, Kluwer Academic Publishers, 2001.
- [4] P. Golani, G. D. Dimou, M. Prakash, and P. A. Beerel, "Design of a high-speed asynchronous turbo decoder," *Proc. of 13th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC'07)*, pp. 49-59, 2007.
- [5] T. E. Murphy, X. Geng, and J. Hammer, "On the control of asynchronous machines with races," *IEEE Transactions on Automatic Control*, vol. 48, no. 6, pp. 1073-1081, Jun. 2003.
- [6] N. Venkatraman and J. Hammer, "On the control of asynchronous sequential machines with infinite cycles," *International Journal of Control*, vol. 79, no. 7, pp. 764-785, Jul. 2006.
- [7] J. Peng and J. Hammer, "Input/output control of asynchronous sequential machines with races," *International Journal of Control*, vol. 83, no. 1, pp. 124-144, Jan. 2010.
- [8] J.-M. Yang and S. W. Kwak, "State feedback control of asynchronous sequential machines with uncontrollable inputs: application to error counters," *Journal of Institute of Control, Robotics and Systems (in Korean)*, vol. 15, no. 10, pp. 967-973, Mar. 2009.
- [9] J.-M. Yang and S. W. Kwak, "Model matching for asynchronous sequential machines with adversarial inputs using state bursts," *International Journal of Control, Automation, and Systems*, vol. 8, no. 5, pp. 985-993, Oct. 2010.
- [10] J.-M. Yang and S. W. Kwak, "Realizing fault-tolerant asynchronous sequential machines using corrective control," *IEEE Transactions on Control Systems Technology*, vol. 18, no. 6, pp. 1457-1463, Nov. 2010.
- [11] Z. Kohavi, *Switching and Finite Automata Theory*, 2nd ed., New York: McGraw-Hill, 1978.
- [12] M. Valencia, M. J. Bellido, J. L. Huertas, A. J. Acosta, and S. Sanchez-Solano, "Modular asynchronous arbiter insensitive to metastability," *IEEE Transactions on Computers*, vol. 44, no. 12, pp. 1456-1461, Dec. 1995.
- [13] N. Hasasneh, I. Bell, and C. Jesshope, "Asynchronous arbiter for micro-threaded chip multiprocessors," *Journal of Systems Architecture*, vol. 53, no. 5-6, pp. 253-262, May-Jun. 2007.
- [14] E. S. Shin, V. J. Mooney III, and G. F. Riley, "Round-robin arbiter design and generation," in *Proceedings of the 15th international symposium on System Synthesis (ISSS'02)*, pp. 243-248, 2002.

양 정 민

1993년 한국과학기술원 전기 및 전자공학과 졸업(공학사). 1995년 한국과학기술원 전기 및 전자공학과 졸업(공학석사). 1999년 한국과학기술원 전기 및 전자공학과 졸업(공학박사). 1999년~2001년 한국전자통신연구원 컴퓨터·소프트웨어 기술연구소 선임연구원. 2001년~현재 대구가톨릭대학교 전자공학과 부교수. 관심분야는 비동기 순차 머신 제어, 실시간 시스템, 결음새 연구.

곽 성 우

1993년 한국과학기술원 전기 및 전자공학과 졸업(공학사). 1995년 한국과학기술원 전기 및 전자공학과 졸업(공학석사). 2000년 한국과학기술원 전기 및 전자공학과 졸업(공학박사). 2000년~2002년 인공위성연구센터 선임연구원, 연구교수. 2003년~현재 계명대 전자공학과 부교수. 관심분야는 위성 탑재 컴퓨터, 실시간 시스템, 비동기 시스템 설계, 내고장성 시스템 설계.