AUTOMATYKA • 2011 • Tom 15 • Zeszyt 2

Magdalena Szymczyk*, Piotr Szymczyk*

Bezpieczeństwo i niezawodność systemów wbudowanych opartych na nowoczesnych mikrokontrolerach**

1. Wprowadzenie

Układy elektroniczne wykorzystują małe pakiety ładunków do reprezentacji bitu informacji. Każde zmiany takiej informacji mogą powodować modyfikacje składowanej informacji. W momencie wystąpienia takiego zjawiska (*SEU – single event upset*), z powodu zakłóceń spowodowanych poprzez uderzenie np. naładowaną cząstką alfa, w układzie elektronicznym może dojść do wystąpienia tzw. przejściowego błędu (*transient error, soft terror*). Błąd ten nazywany jest także miękkim, gdyż nie powoduje zmian w fizycznej strukturze układu. SEU to zjawisko rzadkie, dla większej części użytkowników błędy przejściowe maskowane są przez pojawienie się innych błędnych działań aplikacji (błędy oprogramowania, konfiguracji) i lepiej lub gorzej ten problem jest rozwiązywany [6, 7, 8, 9]. W przypadku wystąpienia SEU w aplikacjach krytycznych, wykorzystywanych do funkcji sterowania pewnym systemem, może mieć on poważny wpływ na jego działanie, w najgorszym przypadku z katastrofą włącznie.

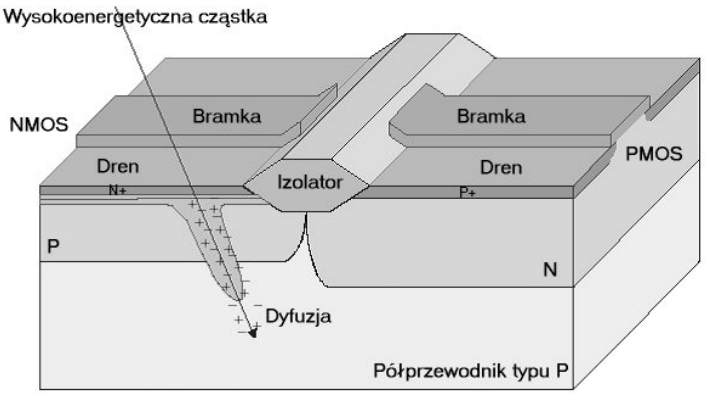
W kilku ostatnich dekadach można zaobserwować ogromne zmniejszanie się rozmiarów układów elektronicznych, co oznacza ogromny przyrost liczby tranzystorów na matrycy, przewidywany przez G. Moore'a. Rozmiary elementów w technologii VLSI ulegają drastycznemu zmniejszeniu, napięcie zasilania ulega zmniejszeniu, zatem niezbędny ładunek potrzebny do prawidłowej zmiany stanu układu również ulega zmniejszeniu. Oczywiście ładunek zakłócający poprawną pracę danego elementu także bardzo się zmniejszył, więc współczesne mikroprocesory stały się bardzo podatne na zakłócenia. Zakłócenia te wynikają ze zdarzenia polegającego na uderzeniu układu z naładowaną cząsteczką pochodzącą z kosmosu (neutron, cząsteczka alfa) (rys. 1). Jeżeli w wyniku tego zjawiska wystąpią pulsacje napięcia, to wówczas mogą się one propagować w układzie i wpłynąć na zmianę stanu

* AGH Akademia Górniczo-Hutnicza, Wydział Elektrotechniki, Automatyki, Informatyki i Elektroniki, Katedra Automatyki

** Projekt został sfinansowany ze środków Narodowego Centrum Nauki 6693/B/T02/2011/40

np. elementu sekwencyjnego i przez to wynik działania programu na takim procesorze będzie niepoprawny.

Dla pewnej grupy aplikacji pojawienie się takich błędów jest niedopuszczalne (może powodować nieobliczalne skutki, łącznie z katastrofami), zatem problem radzenia sobie z tymi błędami jest dość ważny.



Rys. 1. Zjawisko zderzenia wysokoenergetycznej cząstki z tranzystorem [1]

Naukowcy projektujący układy mikroprocesorowe stają przed zadaniem zabezpieczenia takich systemów na odpowiednim poziomie, określonym przez użytkownika. Po pierwsze muszą poznać wpływ tego typu zjawiska na układ, a następnie podjąć odpowiednie działania niwelujące jego wpływ na całość systemu. Konieczny staje się odpowiedni wybór dostępnych technik do zmniejszania ich wpływu przy jednoczesnym minimalizowaniu ich wpływu na działanie całego układu.

2. Metody minimalizacji wpływu błędów SEU na urządzenia mikroprocesorowe

Dostępne są różne sposoby zmniejszania wpływu błędów przejściowych na układy elektroniczne. Można je ogólnie podzielić ze względu na obszar działania w następujący sposób: rozwiązania dotyczące budowy obwodów elektronicznych, rozwiązania realizowane na poziomie procesów czy też architektury układu [2].

2.1. Rozwiązania technologiczne

Oczywiście istnieją metody specjalizowane do zabezpieczania sprzętu przed działaniem promieniowania realizowane w procesie technologicznym. Taką metodą jest wykorzystanie układów z krzemowym izolatorem (ang. SOI). Są to rozwiązania dość kosztowne i raczej nie polecane dla sprzętu ogólnego przeznaczenia [4].

2.2. Ulepszenia dotyczące budowy obwodów elektronicznych

Alternatywnym rozwiązaniem problemu zmniejszania wpływu SER na układy jest projektowanie i wytwarzanie np. pamięci odpornych na działanie promieniowania przez zwiększanie ich pojemności czy też napięcia zasilania. Takie rozwiązania niestety pochłaniają dużo energii zasilania oraz wiążą się ze zwiększoną powierzchnią takiego elementu.

2.3. Rozwiązania dotyczące architektury układu mikroprocesorowego

Rozwiązania tego typu są najczęściej najlepszym rozwiązaniem, ze względu na charakter tego błędu. Rozwiązania bazujące na architekturze można ogólnie podzielić na dwie klasy:

- Rozwiązania dotyczące architektury procesora w mikroskali i są to kody parzystości, SECDED ECC.
- Rozwiązania dotyczące architektury w makroskali – wielokrotnie w mikroprocesorach łatwiej jest zastosować powielenie działania CPU czy też wątków, niż tworzyć dodatkowe układy logiczne potrzebne do działania mikro-rozwiązań.

2.4. Rozwiązania dotyczące architektury układu mikroprocesorowego bazujące na rendundancji

Ogólnie rzecz biorąc, detekcję błędów miękkich (inaczej przejściowych) można zrealizować przy użyciu redundancji wykonania [2].

Redundancja wykonania może zostać podzielona następnie na trzy klasy:

- Redundancję przestrzenną
- Redundancję czasową
- Redundancję informacji

Replikacja przestrzenna oparta jest na wykonaniu jednej aplikacji na wielu identycznych platformach sprzętowych i porównaniu wyników w celu określenia błędnego działania. Charakteryzuje się ona wysokim stopniem detekcji błędów przy małym wpływie na wydajność pracy, ale przy znacznym koszcie wykorzystanego sprzętu.

Replikacja czasowa w przeciwieństwie do przestrzennej wykonuje tę sama aplikację wielokrotnie na tym samym sprzęcie i porównuje wyniki. Koszt tej realizacji jest niski przy wysokim współczynniku detekcji błędów, niestety wpływ na wydajność jest ogromny przy dość dużej komplikacji budowy układu.

Redundancja informacji charakteryzuje się wykorzystaniem pewnych elementów w postaci dodatkowych bitów sprawdzających integralność danych (np. kod parzystości dla pamięci). Ta forma sprawdza się dla układów przechowujących dane.

2.5. Wykrywanie błędów za pomocą wykonania redundancyjnego

W przemyśle realizuje się najczęściej dwa typy schematów redundancji wykonania. Pierwszy to tzw. *lockstepping* – charakteryzujący się sprawdzaniem stanu wykonania kolejnych kopii w kolejnych cyklach pracy procesora. Stany te muszą być identyczne, w przeciwnym przypadku wykrywany jest błąd [2].

Druga metoda tzw. *redundant multithreading* (RMT) – polega na porównywaniu wyjścia zakończonych instrukcji. Szczególnie istotną informacją jest to, że stan wewnętrzny poszczególnych wątków może być różny, gdy nastąpi błąd. Ten sposób nie nakłada warunku synchronizacji stanu po każdym cyklu, zatem może on zostać zrealizowany na wiele sposobów. Metoda ta w porównaniu z poprzednią charakteryzuje się większą elastycznością, jeśli chodzi sposób realizacji, i może zostać zaimplementowana na każdej architekturze realizującej wielowątkowość. Niezależne wątki mogą być realizowane na osobnym sprzęcie, na osobnym rdzeniu procesora czy też specjalizowanym elemencie sprawdzającym. Porównywanie stanów w tej metodzie jest łatwiejsze, gdyż realizowane jest na poziomie architektury procesora, a nie mikroarchitektury, zatem stan maszyny stanów realizującej konkretny program jest dostępny przez programistę.

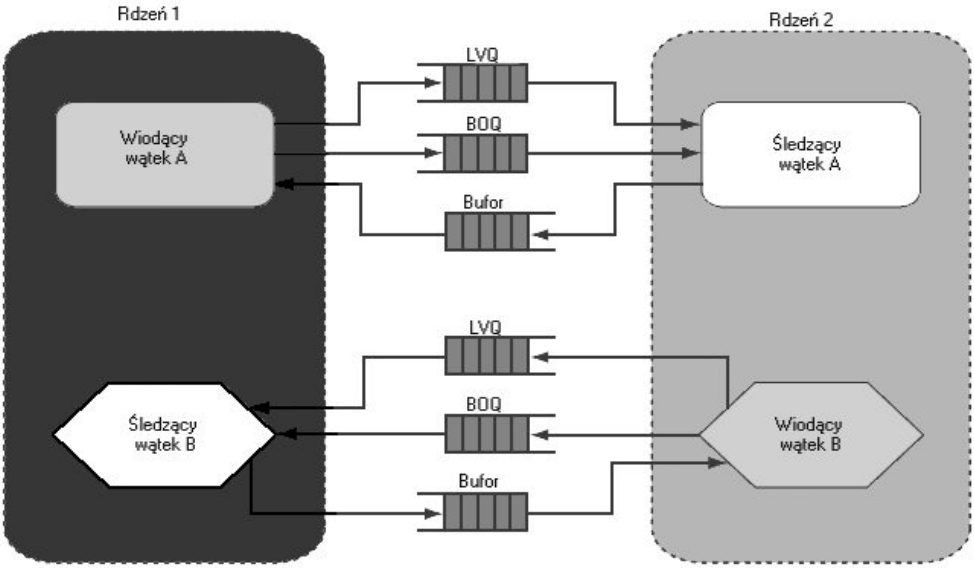
2.6. Realizacja metody RMT na procesorze SMT (*Simultaneous Multithreaded Processor*)

Technika SMT [3] – jednoczesna wielowątkowość – pozwala na niskopoziomowe dzielenie zasobów nowoczesnego superskalarnego procesora pomiędzy wiele wątków. W czasie jednego cyklu procesora wykonywanych jest wiele instrukcji pochodzących z różnych wątków, a przykładami takich procesorów są Intel Atom czy też Intel Pentium 4. Tego typu procesor można w łatwy sposób zmodyfikować do celów detekcji błędów, poprzez wykonanie dwóch redundantnych kopii każdego wątku w osobnych kontekstach. Procesor wzbogacony o tę technikę określa się skrótem SRT (*Simultaneous Redundantly Threaded*). W procesorze tym jeden z dwóch redundantnych wątków, uruchamiany jest nieco wcześniej przed drugim, zatem pierwszy określa się jako wiodący, drugi jako śledzący. W celu zaimplementowania techniki SRT na procesorze SMT dodaje się dwa kluczowe mechanizmy: replikacji wejścia oraz porównania wyjścia. Zadaniem mechanizmu pierwszego jest zagwarantowanie, że oba wątki będą miały te same wartości wejściowe, drugi sprawdza, czy wyjście z nich jest identyczne. W przypadku stwierdzenia rozbieżności mechanizm sprawdzania wyjścia ma za zadanie zasygnalizować błąd i ewentualnie podjąć akcję wychodzenia z tego stanu. Sfera replikacji (granica logiczna systemu, w której dokonuje się redundantnego wykonania) obejmuje swym działaniem w SRT zazwyczaj (jedna z możliwości): potoki, bloki rejestrów, a wyklucza cache poziomu L1 dla danych oraz instrukcji. Dla tych układów wyłączonych ze sfery replikacji należy dokonać replikacji operacji zwią-

zanych z pobieraniem danych oraz pobierania instrukcji z cache. Dokonuje się tego przy użyciu specjalnego bufora realizującego kolejkę FIFO dla pobieranych wartości. Każdy z wiodących wątków, gdy zakończy operację pobierania z cache, zapisuje dane oraz adres do bufora *load value queue* (LVQ*)*. Operacja pobierania wątku śledzącego weryfikuje adres pobierany i następnie czyta wartości z kolejki. Bufor ten znajduje się poza obszarem replikacji, zatem należy go chronić przed błędami np. z pomocą kodu ECC. Takie działanie zabezpiecza SRT przed zewnętrznymi zmianami w pamięci np. w wyniku działania urządzeń I/O. Można dodatkowo zwiększyć wydajność procesora SRT przez wykorzystanie jednego z wątków do przyspieszenia działania drugiego, przy operacjach skoku warunkowego czy na pamięci cache. Jedno z możliwych rozwiązań oparte jest na wykorzystaniu kolejnej kolejki *branch outcome queue* (BOQ). Mechanizm ten zabezpiecza przed podjęciem wykonania nieprawidłowych ścieżek w operacjach skoku warunkowego w wątku śledzącym. Wiodący wątek realizujący wcześniej tę instrukcję do kolejki dostarcza już konkretne informacje dotyczące wartości rejestru PC oraz adres skoku dla niej. Zatem watek śledzący nie musi już wykonywać pewnych operacji i podąża dokładnie tę samą ścieżką jak wątek wiodący. W czasie wykonania programu ten sposób działania powoduje, że mniejsza ilość zasobów procesora jest wykorzystywana (potok, przepustowość obliczeniowa), co skutkuje lepszą wydajnością pracy procesora SRT. Niestety zaimplementowanie techniki SRT na procesorze SMT skutkuje 32-procentową degradacją wydajności dla wersji bez żadnych redundantnych kopii (jeden wątek). Jednakże zastosowanie SRT na procesorze wykonującym dwie redundantne kopie tego samego programu powoduje wzrost wydajności o 11% (bez operacji replikacji danych i wyjściowego porównania). Wynika to z zastosowania kolejki LVQ, po pierwsze dzięki jej zastosowaniu watek śledzący nigdy nie pobierze wartości z pamięci cache, której w nim nie ma (*cache miss*), po drugie wątki nie współzawodniczą w dostępie do pamięci cache.

2.7. Realizacja techniki RMT na architekturze wielordzeniowej

W tej technice [5] rozszerza się zastosowanie STR do coraz liczniejszej grupy procesorów wielordzeniowych i określa się tę technikę mianem *chip-level redundantly threaded* (CRT). Idea działania tego procesora CRT jest taka sama jak powyższego, z tą różnicą, że wątki główny i śledzący wykonywane są na osobnych rdzeniach, jak pokazano na rysunku 2. Procesory CRT mają pewną przewagę nad procesorami implementującymi technikę lockstepping. W tej technice procesory wszystkie sygnały wyjściowe muszą porównywać, także te, wynikające z dostępu do pamięci cache. Jest to bardzo niekorzystne zjawisko dla wydajności procesora, gdyż sytuacja braku danych w pamięci cache (cache miss) to ścieżka krytyczna dla wykonania programu. Po drugie – umieszczenie dwóch wątków na dwóch różnych rdzeniach zwiększa całkowitą przepustowość mikroprocesora. Dodatkowo w metodzie CRT nie jest konieczna synchronizacja wątków, o ile odstęp pomiędzy nimi nie wzbudzi licznika watchdoga.



Rys. 2. Diagram blokowy implementacji CRT przy użyciu dwóch rdzeni procesora

3. Wnioski

W systemach mikroprocesorowych odpornych na błędy, wykonanie redundantne opiera się na wykonaniu identycznych kopii programu i porównaniu wyników działania. W przypadku różnic sygnalizowane jest uszkodzenie i podejmowana jest akcja próby powrotu do normalnej pracy. W systemach mikroprocesorowych najbardziej powszechne podejścia do redundantnego wykonania to metoda lockstepping wykonywana po każdym cyklu pracy procesora i znana od wielu dziesięcioleci. Metoda RMT jest metodą nową, wprowadzoną w ostatnim dziesięcioleciu. W tej metodzie szczególną uwagę należy zwrócić na replikację wejść, gdyż redundantne konteksty mogą mieć różne stany w tym samym cyklu. Daje ona możliwość efektywnego wykorzystania zasobów procesorów SMT na drobnoziarnistym poziomie pomiędzy wątkami.

Metoda ta charakteryzuje się dużo lepszym pokryciem błędów przejściowych niż stosowane komercyjnie układy z replikacją statyczną realizowaną za pomocą sprzętu. Procesor typu CRT (wielowątkowy) jest około 13% wydajniejszy niż 2 jednostki CPU realizujące metodę lockstepping. Technika ta jest zatem interesująca ze względu na coraz powszechniejsze procesory wielordzeniowe. Realizowanie techniki RMT dodatkowo ma tę własność, że może być ona realizowana opcjonalnie.

Literatura

[1] Mastipuram R., Wee E.: *Soft Errors' Impact on System Reliability*. EDN, 2004, http://www.edn.com/article/CA454636.html.

[2] Mukherjee S.: *Architecture design for soft errors*. Elsevier, 2008.
[3] Mukherjee S., Kontz M., Reinhardt S.: *Detailed Design and Evaluation of Redundant Multithreading Alternatives*. Proc. 29th Annual Int'l Symp. on Computer Architecture (ISCA), 2002 99–110.
[4] Mukherjee S., Emer J. reinhardt S.: *The Soft Error Problem: An Architectural PerspectiveII*. Proc. 11th Int'l Symp on High-Performance Computer architecture (HPCA) 2005.
[5] Reinhardt S., Mukherjee S.: *Transient Fault Detection via Simultaneous Multithreading*. Proc. 27th Annual Int'l Symp. on Computer Architecture (ISCA), 2000.
[6] Szymczyk M., Szymczyk P.: *Programowe techniki zwiększania odporności na błędy przemijające w systemach czasu rzeczywistego*. Praca zbiorowa pod redakcją Z. Zielińskiego: „Systemy czasu rzeczywistego. Postęp badań i zastosowania", WKŁ, Warszawa 2009, 243–254.
[7] Szymczyk M., Szymczyk P.: *Techniki programistyczne stosowane w niezawodnych komputerowych systemach sterowania*. Automatyka (półrocznik AGH), t. 13, z. 2, 2009, 601–606.
[8] Szymczyk M., Szymczyk P.: *Detekcja błędów on-line w komputerowych systemach sterowania za pomocą procesora monitorującego*. Pomiary Automatyka Robotyka, 2, 2010, 689–695.
[9] Szymczyk M., Szymczyk P.: *Zastosowanie mechanizmu zegarowego do wykrywania anomalii zachowania komputerowego systemu sterowania*. Pomiary Automatyka Robotyka, 2, 2010, 684–688.