MICROCODE-CONTROLLED RAM BIST

Vykydal Lukáš
Master Degree Programme (2), FEEC BUT
E-mail: xvykyd04@stud.feec.vutbr.cz

Supervised by: Kubíček Michal
E-mail: kubicek@feec.vutbr.cz

Abstract: This paper deals with memory testing principles, focusing mainly on March algorithms. It describes their usage on word based memories. In second part it proposes small programmable BIST controller that can be used in digital circuits.

Keywords: memory testing, March algorithms, memory BIST, counters

1 ÚVOD

Polovodičové paměti zabírají stále větší část plochy integrovaných obvodů. S rostoucí velikostí roste i čas nutný k otestování pamětí, který je již při testu z externího ATE (Automatic Test Equipment) neúnosný. Požadavek na zkrácení doby testu vedl k návrhu vestavných testovacích bloků (BIST - Built In Self Test). Jejich využitím došlo ke zrychlení testů.

S dalším vývojem jsou ovšem odhalovány a modelovány další možné defekty na paměťových buňkách. Cílem je proto mít k dispozici programovatelný BIST blok, který umožní testovat paměť na provozní rychlosti.

2 TESTY PAMĚTÍ

Plný test paměti, při kterém je každá paměťová buňka přečtena ve všech možných kombinacích ostatních paměťových buněk, je jistě přespříliš časově náročný. Vyloučením nepravděpodobných chyb a optimalizací vznikly testovací algoritmy různých časových složitostí. Krátký seznam testovacích algoritmů je v tabulce 1. SaF (Stuck at Fault) je defekt, kdy je paměťová buňka trvale v 0 nebo 1. TF (Transition Fault) nazýváme chybu, kdy jedna z logických hodnot nelze do paměti zapsat. CF (Coupling Fault) jsou vázané chyby mezi buňkami.

Algoritmus	Složitost	Defekty
Walking 0-1	$O(n)$	SaF
GALPAT	$O(n^2)$	SaF, TF, CF
Sliding diagonal	$O(n^{3/2})$	SaF, TF, CF
March algoritmy	$O(n)$	Dle algoritmu

Tabulka 1: Složitosti algoritmů

Zápis	Význam
$w0/w1$	Zápis na aktuální paměťovou pozici
$r0/r1$	Porovnání aktuální paměťové pozice
$\Updownarrow(\ldots)$	Skupina prováděných operací
$\Uparrow(\ldots)$	... ve vzestupném pořadí
$\Downarrow(\ldots)$	... ve sestupném pořadí

Tabulka 2: Gramatika March testů

March algoritmy jsou skupinou algoritmů pro testování pamětí s lineární časovou složitostí. Příkladem budiž algoritmus MATS+ (Modified Algorithmic Test Sequence) formálně zapsaný ve výrazu (1) pomocí gramatiky definované v tabulce 2. Jde o nejjednodušší March test. Do paměti je nejdříve na všechny pozice zapsána log. 0. Následně jsou data zkontrolována a nahrazena log. 1. Ve třetím kroce je opět obsah paměti porovnán s log. 1.

$$\Updownarrow\ (w0) \Uparrow\ (r0,w1) \Downarrow\ (r1,w0) \tag{1}$$

S pomocí vhodné kombinace základních operací se dá dosáhnout velmi dobrého pokrytí jednobitových chyb a omezené skupiny vícebitových. Příkladem buď algoritmus March SS zapsaný ve výrazu (2). Tento March test pokryje navíc chyby DRF (Destructive Read) a dDRF (Deceptive Destructive Read - přečtená data jsou správná, ale dojde k poškození dat v paměti) pomocí zdvojeného čtení a WDF (Write Disturb Fault - zápisem stejné hodnoty jako je uložena v paměti dojde k poškození uložené hodnoty) pomocí opakovaného zápisu stejné hodnoty.

$$\Updownarrow\ (w0) \Uparrow\ (r0,r0,w0,r0,w1) \Uparrow\ (r1,r1,w1,r1,w0) \Downarrow\ (r0,r0,w0,r0,w1) \Downarrow\ (r1,r1,w1,r1,w0) \Updownarrow\ (r0) \tag{2}$$

K pokrytí defektů mezi sousedními buňkami se využije šachovnicový vzor zapsaný do paměti. Pro jeho generování je nutné znát fyzické rozložení paměti.

2.1 March algoritmy a paměť s přístupem po slovech

Standardní March testy jsou definované nad pamětí adresovanou po bitech. V reálném použití se vyskytují paměti adresované po slovech různých délek. March algoritmy můžeme tedy aplikovat dvěma způsoby:

- Pro test vybrat skupinu slov a jejich inverzí, které budeme využívat místo hodnoty 0,1. Tato slova můžeme ve vhodných místech během testu střídat nebo nechat celý test proběhnout několikrát.
- Využít SMARCH (Serial March) testu. Jde o úpravu March algoritmů, kdy se k paměťovému slovu přistoupí jako k posuvnému registru, kdy na vstupu je hodnota zapisovaná do paměti a provádí se kontrola výstupní hodnoty. Každá operace SMARCH algoritmu tady trvá n cyklů, kde n je délkou slova testované paměti.

3 BIST KONTROLÉR

Struktura navrhovaného kontroléru je na obrázku 1. BIST kontrolér je rozdělen do dvou částí. V první je obsažená LUT (Lookup Table) s mikrokódem a implementace mikrosekvenceru. Druhá část slouží k přístupu k paměti a generování testovacích datových slov. Cílem tohoto rozdělení je možnost přepoužít mikrosekvencer pro více pamětí na čipu.

Pro implementaci šachovnicového vzoru jsou adresní čítače BIST kontroléru rozděleny na řádkový a sloupcový. Složením jejich výstupů dohromady vznikne celá adresa pro paměť. To také umožní přistupovat k paměti po sloupcích nebo po řádcích a zvýšit pokrytí vázaných chyb.

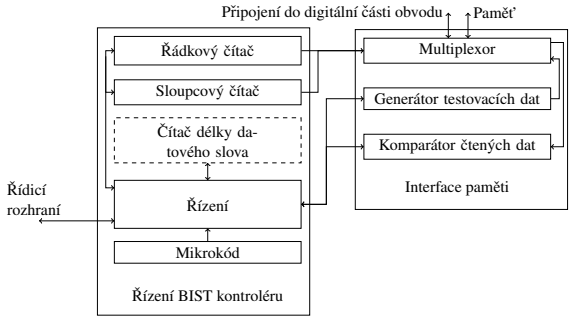
Navrhovaný BIST kontrolér také bude obsahovat sériové rozhraní pro přístup k paměti za účelem diagnózy.

3.1 Optimalizace velikosti

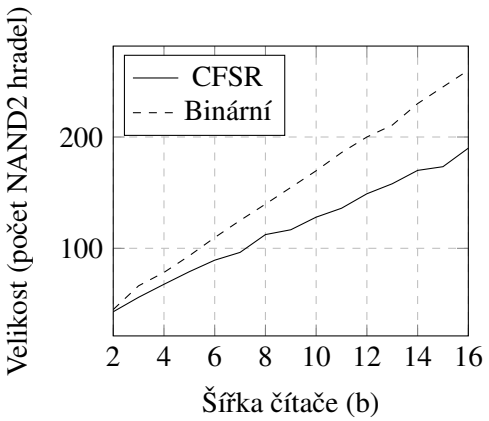
Pro adresaci přístupu k paměťovým buňkám je potřeba využít čítač, který pokryje všech 2^n možných kombinací adresního slova. Typicky se pro toto adresování využívá binární čítač, který přistupuje k adresním pozicím v „přirozeném“ pořadí.

Dle článku [2] lze pozorovat určité „stupně volnosti“ v definici March algoritmů při zachování jejich pokrytí chyb. Pro nás je zajímavý stupeň volnosti 1, který umožňuje využít libovolné reverzibilní posloupnosti pro adresaci paměti March algoritmem.

Pro porovnání byly zvoleny čítače binární a CFSR (Complete Feedback Shift Register - jde o čítač LFSR (Linear Feedback Shift Register) s vnuceným stavem `(others => '0');` jeho reverzibilita vychází z reverzibility LFSR čítačů [3]). Jako kritérium pro velikost výsledné realizace čítače na čipu bylo zvoleno množství NAND2 hradel, které by pokrylo plochu zabranou tímto čítačem. Výsledky porovnání jsou vidět v grafu 2. Z tohoto porovnání je zřejmé, že čítače CFSR zaberou v tomto MBIST kontroléru značně menší plochu oproti běžným binárním čítačům při šířce čítače větší než 2 bity.



Obrázek 1: Bloková struktura MBIST kontroléru



Obrázek 2: Závislost velikosti čítače jeho šířce

4 ZÁVĚR

Porovnání klasických metod testování polovodičových pamětí ukázalo, proč jsou March algoritmy pro test pamětí velmi populární. Po jejich rozšíření směrem k SMARCH nebo k většímu množství testovacích slov je možné je použít na pamětech s přístupem po slovech a využít šachovnicový vzor dat. Pro úsporu místa na čipu navrhuji použít čítače typu CFSR pro generování adresních posloupností.

PODĚKOVÁNÍ

Děkuji společnosti ON Design Czech s.r.o. a jejím zaměstnancům, zejména Ing. Miroslavu Kaššovi za pomoc, rady a možnost zpracovávat práci v prostorách firmy.

REFERENCE

[1] S. Hamdioui, A. J. van de Goor and M. Rodgers, *March SS: A Test for All Static Simple RAM Faults*, Proceedings of the 2002 IEEE International Workshop on Memory Technology, Design and Testing (MTDT2002), 2002, pp. 95-100. Dostupné z URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1029769>

[2] D. Niggemeyer, M. Redeker and J. Otterstedt, *Integration of non-classical faults in standard March tests*, Proceedings. International Workshop on Memory Technology, Design and Testing (Cat. No.98TB100236), San Jose, CA, 1998, pp. 91-96. Dostupné z URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=705953&isnumber=15293>

[3] Dhingra Sachin. *Comparison of LFSR and CA for BIST*. Auburn University, USA Dostupné z URL: <http://www.eng.auburn.edu/~agrawvd/COURSE/E7250_05/REPORTS_TERM/Dhingra_LFSR.pdf>