

رسالة محمد



دانشکده فنی

پایان نامه کارشناسی ارشد

طراحی قابل سنتز تحلیل گر تعمیر داخلی برای

حافظه های بر مبنای کلمه

از:

شاهین خدادادی

استاد راهنما:

دکتر راهبه نیارکی اصلی

شهریور ۱۳۹۲

تقدیم به:

پدر و مادر عزیزم

که در هنگام سختی ها و دشواری های زندگی، همواره یاری دلسوز و خداکار برایم بوده اند. پشتیبانی و رهنمودهای ایشان، همواره امید به موفقیت را در من زنده نگه داشته است.

شکر و قدردانی:

از استاد فرهیخته و فرزانه ام سرکار خانم دکتر راهبه نیارکی اصلی به خاطر زحمات فراوانشان در به ثمر رسیدن این پایان نامه کمال شکر را دارم.
راهنمایی های سازنده ایشان سبب بهوار شدن مسیر این پژوهش بوده است.

چکیده

طراحی قابل سنتز تحلیل گر تعمیر داخلی برای حافظه‌های کلمه‌گرا

شاهین خدادادی

با پیشرفت تکنولوژی مدارات VLSI و کاهش ابعاد مدارهای الکترونیکی شاهد افزایش میزان نقص‌ها یا خطاهای ادوات به‌خصوص در حافظه‌ها هستیم. بروز خطا در ادوات الکترونیکی سبب کاهش بهره‌دهی مدارات و کاهش قابلیت اطمینان خواهد شد. در این راستا تحقیقات زیادی برای طراحی مدارهای تعمیر درون‌ساخته صورت گرفته تا از کاهش بهره‌دهی و قابلیت اطمینان ادوات جلوگیری شود. با توجه به بالا بودن نیاز مدارات دیجیتال به ذخیره‌ی داده‌ها، حافظه‌های متعددی برای ذخیره‌سازی اطلاعات استفاده می‌شوند. حافظه‌ها با توجه به چینش سلول‌های آن به دو دسته‌ی بیت‌گرا و کلمه‌گرا تقسیم‌بندی می‌شوند.

در روش‌های قدیمی‌تر، از ابزارهای آزمون خودکار (ATE) و الگوریتم‌های خارج تراشه استفاده می‌شد. اما افزایش مداوم پیچیدگی هسته‌های SOC، سبب دشواری استفاده از ATE شده است. همین عامل نیاز به روش‌های خودآزمون درون‌ساخته (BIST) و مدار خودتعمیر درون‌ساخته (BIRA) را تشدید کرد. در روش‌های امروزی، بخش‌هایی از حافظه به عنوان افزونه‌های کمکی در نظر گرفته می‌شوند و به منظور جایگزینی سلول‌های معیوب حافظه به کار می‌روند. مهمترین بخش یک مدار خودتعمیر درون‌ساخته، تحلیل‌گر افزونه‌ی درون‌ساخته (BIRA) نام دارد که پس از دریافت اطلاعات از مدار BIST به تحلیل آدرس‌های خطا پرداخته و نتیجه‌ی تحلیل سبب تغییر مسیر داده به سمت افزونه‌ها خواهد شد. در این پایان‌نامه به طراحی دو مدار تحلیل‌گر افزونه‌ی درون‌ساخته برای تعمیر حافظه‌های کلمه‌گرا پرداخته شده است. طرح پیشنهادی اول برای تعمیر خطاهای چند بیتی ارائه شده است. این طرح با پذیرش هزینه‌ای بابت افزونه‌های بیشتر، دستیابی به نرخ تعمیر بهینه و تعمیر بلادرنگ را ممکن ساخته است. طرح دوم با استفاده از خواص حافظه‌ی بیت‌گرا و کلمه‌گرا و تکنیک تبدیل آدرس، برای تعمیر خطاهای تک‌بیتی ارائه شده است که ضمن تحلیل بلادرنگ، به نرخ تعمیر بالایی دست پیدا می‌کند. دو روش پیشنهادی علاوه بر مزایای فوق، فضای کمتری از تراشه را به منظور پیاده‌سازی اشغال می‌کنند. به طوری که طرح پیشنهادی اول به ترتیب $78/8\%$ و $63/6\%$ نسبت به روش‌های EESP و 1-D Bitmap بهبود یافته است. همچنین طرح پیشنهادی دوم دارای $59/5\%$ و $30/4\%$ بهبود در فضای اشغالی می‌باشد.

واژه‌های کلیدی:

حافظه‌های جاسازی شده، حافظه‌های کلمه‌گرا، مدار خودآزمون درون‌ساخته، مدار خودتعمیر درون‌ساخته، تحلیل‌گر افزونه‌ی درون‌ساخته

فهرست مطالب

فصل ۱: مقدمه

۲-۱	مقدمه	۲
۲-۱	مفاهیم اولیه	۵
۱-۲-۱	خطای نرم و خطای سخت	۵
۲-۲-۱	ECC	۷
۳-۲-۱	افزونه	۷
۴-۲-۱	بهره‌دهی	۸
۳-۱	ساختار حافظه	۱۱
۱-۳-۱	حافظه‌ی بیت‌گرا	۱۱
۲-۳-۱	حافظه‌ی کلمه‌گرا	۱۲
۴-۱	تعمیر حافظه با استفاده از افزونه‌ها	۱۴
۱-۴-۱	تعمیر حافظه‌های بیت‌گرا با استفاده از افزونه‌ها	۱۴
۲-۴-۱	تعمیر حافظه‌های کلمه‌گرا با استفاده از افزونه‌ها	۱۴
۵-۱	هدف و ساختار پایان‌نامه	۱۹

فصل ۲: تعمیر حافظه‌های کلمه‌گرا

۱-۲	خودتعمیری حافظه‌های جاسازی شده	۲۱
۱-۱-۲	تحلیل‌گر افزونه‌ی درون‌ساخته	۲۱
۲-۱-۲	پیکربندی مجدد	۲۲
۲-۲	عوامل مهم در تحلیل خطاها	۲۶
۳-۲	روش‌های مختلف تحلیل خطاها	۲۷
۱-۳-۲	طراحی بدون استفاده از بیت‌مپ	۲۷
۲-۳-۲	طراحی با استفاده از بیت‌مپ	۳۰

فصل ۳: طراحی تحلیل‌گر افزونه‌ی درون‌ساخته برای تعمیر خطاهای چند بیتی حافظه‌های کلمه‌گرا

۴۰	۱-۳ مقدمه.....
۴۱	۲-۳ معرفی الگوریتم CRESTA.....
۴۳	۳-۳ معرفی روش پیشنهادی برای تعمیر حافظه‌های کلمه‌گرا.....
۵۰	۴-۳ طراحی مدار Wordy-R-CRESTA.....
۵۳	۵-۳ نتایج شبیه‌سازی Wordy-R-CRESTA.....
۵۵	۶-۳ پیاده‌سازی Wordy-R-CRESTA.....
۵۷	۷-۳ مقایسه.....
۵۷	۱-۷-۳ سرعت تحلیل.....
۵۸	۲-۷-۳ نرخ تعمیر.....
۶۰	۳-۷-۳ فضای اشغالی.....

فصل ۴: طراحی تحلیل‌گر افزونه‌ی درون‌ساخته برای تعمیر خطاهای تک بیتی حافظه‌های کلمه‌گرا

۶۴	۱-۴ مقدمه.....
۶۴	۲-۴ معرفی روش پیشنهادی BIRAی متقارن توسعه یافته برای حافظه‌های کلمه‌گرا.....
۶۸	۳-۴ طراحی تحلیل‌گر BIRAی متقارن توسعه یافته برای حافظه‌های کلمه‌گرا.....
۷۰	۴-۴ نتایج شبیه‌سازی BIRAی متقارن توسعه یافته.....
۷۲	۵-۴ نتایج پیاده‌سازی BIRAی متقارن توسعه یافته.....
۷۴	۶-۴ مقایسه.....
۷۴	۱-۶-۴ فضای اشغالی.....
۷۷	۲-۶-۴ سرعت تحلیل.....
۷۹	۳-۶-۴ نرخ تعمیر.....

فصل ۵: جمع‌بندی و پیشنهادات

۸۱	۱-۵ جمع‌بندی و نتیجه‌گیری.....
۸۲	۲-۵ پیشنهادها.....

۸۳

۸۹

۹۴

مراجع

پیوست‌ها

پیوست الف

پیوست ب

فهرست جداول

- جدول (۱-۱) تعداد ترانزیستورهای مجتمع طی سال‌های متفاوت [۱]..... ۲
- جدول (۱-۳) گزارش میزان استفاده Wordy-R-CRESTA از ادوات منطقی XC5VLX50T برای یک حافظه $۸ \times ۸ \times ۴$ ۵۶
- جدول (۲-۳) گزارش نتایج زمانی شبیه‌سازی مدار پیشنهادی Wordy-R-CRESTA..... ۵۸
- جدول (۳-۳) تعداد ثبات‌های مورد نیاز برای پیاده‌سازی Wordy-R-CRESTA در مقابل روش‌های کلمه‌گرای دیگر برای یک حافظه $۷ \times ۶ \times ۳۲$ ۶۱
- جدول (۴-۳) نسبت افزونه‌ی مورد نیاز به ساین حافظه‌ی $۶ \times ۷ \times ۳۲$ در روش‌های EESP، 1-D Bitmap و روش پیشنهادی Wordy-R-CRESTA..... ۶۲
- جدول (۱-۴) گزارش میزان استفاده‌ی BIRA مقارن توسعه یافته از ادوات منطقی XC5VLX50T برای یک حافظه $۸ \times ۲ \times ۴$ ۷۳
- جدول (۲-۴) تعداد عناصر منطقی مورد نیاز برای پیاده‌سازی مدار BIRA مقارن توسعه یافته..... ۷۳
- جدول (۳-۴) تعداد ثبات‌های مورد نیاز برای پیاده‌سازی BIRA مقارن توسعه یافته در مقابل روش‌های دیگر برای یک حافظه $۶ \times ۷ \times ۳۲$ ۷۵
- جدول (۴-۴) مقایسه‌ی تعداد ثبات‌های مورد نیاز دو روش BIRA مقارن توسعه یافته و Wordy-R-CRESTA برای یک حافظه $۶ \times ۷ \times ۳۲$ ۷۶
- جدول (۵-۴) نسبت افزونه‌ی مورد نیاز دو روش پیشنهادی به ساین حافظه برای یک حافظه $۶ \times ۷ \times ۳۲$ ۷۷
- جدول (۶-۴) گزارش نتایج زمانی شبیه‌سازی مدار پیشنهادی BIRA مقارن توسعه یافته..... ۷۸

فهرست شکل‌ها

- شکل (۱-۱) روند افزایش پیچیدگی حافظه [۲]..... ۳
- شکل (۲-۱) خطای نرم و عوامل آن [۹]..... ۵
- شکل (۳-۱) تکنیک‌های تعمیر حافظه‌ها هنگام وقوع خطای نرم و سخت: افزونه و ECC [۹]..... ۶
- شکل (۴-۱) وجود خطا در عناصر حافظه [۹]..... ۹
- شکل (۵-۱) بهبود بهره‌دهی با استفاده از جایگزینی بر اساس توزیع پواسون [۹]..... ۱۰
- شکل (۶-۱) ساختار بلوکی حافظه‌ی بیت‌گرا [۲]..... ۱۲
- شکل (۷-۱) ساختار بلوکی حافظه‌ی کلمه‌گرا [۲]..... ۱۳
- شکل (۸-۱) (الف) جایگزینی فقط با افزونه‌ی ردیف [۲۱]، (ب) دیاگرام مفهومی جایگزینی ردیف..... ۱۵
- شکل (۹-۱) (الف) جایگزینی فقط با افزونه‌ی ستون [۲۱]، (ب) دیاگرام مفهومی جایگزینی ستون..... ۱۶
- شکل (۱۰-۱) (الف) جایگزینی با I/O [۲۱]، (ب) دیاگرام مفهومی جایگزینی I/O..... ۱۷
- شکل (۱۱-۱) جایگزینی با ردیف و ستون..... ۱۸
- شکل (۱-۲) دیاگرام بلوکی یک حافظه‌ی مجهز به مدار خودتعمیر..... ۲۲
- شکل (۲-۲) جایگزینی با استفاده از دیکدر قابل برنامه‌ریزی و حذف به روش مستقیم [۹]..... ۲۳
- شکل (۳-۲) جایگزینی با استفاده از روش مقایسه آدرس و حذف به روش غیر مستقیم [۹]..... ۲۴
- شکل (۴-۲) جایگزینی با استفاده از روش شیفت [۹]..... ۲۵
- شکل (۵-۲) درخت جستجوی باینری..... ۲۸
- شکل (۶-۲) (الف) یک نمونه حافظه‌ی معیوب، (ب) استفاده از استراتژی CCRR برای تعمیر حافظه،
(ج) استفاده از استراتژی CRRC برای تعمیر حافظه [۳۸]..... ۲۹
- شکل (۷-۲) سازمان‌دهی افزونه‌ها با روش EESP و تعمیر حافظه [۳۰]..... ۳۱
- شکل (۸-۲) نحوه‌ی سازمان‌دهی افزونه‌ها در الگوریتم بیت‌مپ یک بعدی برای افزونه‌های دو بعدی [۴۲]..... ۳۲
- شکل (۹-۲) نمودار بلوکی روش بیت‌مپ یک بعدی [۴۲]..... ۳۲

- شکل (۱۰-۲) دیاگرام بلوکی BISR برای حافظه‌ی فلش [۴۳]..... ۳۳
- شکل (۱۱-۲) نمای کلی روش افزونه‌ی سه بعدی [۴۴]..... ۳۵
- شکل (۱۲-۲) (الف) مثالی از یک حافظه‌ی کلمه‌گرای معیوب، (ب) پر شدن بیت‌مپ و استفاده از افزونه‌ی کلمه، (ج) پر شدن بیت‌مپ و استفاده از روش LRM [۴۴]..... ۳۶
- شکل (۱۳-۲) بلوک دیاگرام مدار ارائه شده در [۴۵]..... ۳۷
- شکل (۱۴-۲) مثالی برای الگوریتم موجود در [۴۵]..... ۳۸
- شکل (۱-۳) یک نمونه حافظه‌ی ۸×۸ معیوب با دو افزونه‌ی سطر و دو افزونه‌ی ستون..... ۴۲
- شکل (۲-۳) استفاده از استراتژی R-C-R-C برای تعمیر حافظه‌ی شکل (۱-۳)..... ۴۲
- شکل (۳-۳) استفاده از استراتژی R-C-C-R برای تعمیر حافظه‌ی شکل (۱-۳)..... ۴۳
- شکل (۴-۳) درخت جستجوی دودویی پیشنهادی..... ۴۴
- شکل (۵-۳) روندنمای الگوریتم پیشنهادی..... ۴۵
- شکل (۶-۳) حافظه‌ی کلمه‌گرای معیوب با دو افزونه‌ی ردیف و دو افزونه‌ی کلمه..... ۴۶
- شکل (۷-۳) مراحل تحلیل خطا و تعمیر حافظه‌ی کلمه‌گرای معیوب شکل (۶-۳) توسط الگوریتم Wordy-R-CRESTA..... ۴۷
- شکل (۸-۳) استراتژی موفق برای تعمیر حافظه‌ی کلمه‌گرای شکل (۶-۳)..... ۴۹
- شکل (۹-۳) مدار RTL سلول تحلیل‌گر پیشنهادی Wordy-R-CRESTA..... ۵۰
- شکل (۱۰-۳) زیرتحلیل‌گر RRWW برای تحلیل‌گر پیشنهادی Wordy-R-CRESTA..... ۵۱
- شکل (۱۱-۳) دیاگرام بلوکی تحلیل‌گر پیشنهادی Wordy-R-CRESTA..... ۵۲
- شکل (۱۲-۳) نتیجه‌ی شبیه‌سازی الگوریتم پیشنهادی Wordy-R-CRESTA با استفاده از آدرس‌های خطای شکل (۶-۳)..... ۵۴
- شکل (۱۳-۳) نتیجه‌ی مشاهده شده پس از پیاده‌سازی Wordy-R-CRESTA بر روی تراشه‌ی XC5VLX50T..... ۵۶
- شکل (۱۴-۳) تعداد سیکل‌های مورد نیاز تحلیل‌گرهای مختلف بر حسب تعداد خطا برای شروع تحلیل..... ۵۷

- شکل (۳-۱۵) مقایسه‌ی نرخ تعمیر روش پیشنهادی Wortdy-R-CRESTA به ازای آرایش مختلف افزونه‌ها در یک حافظه‌ی ۸۱۹۲×۶۴..... ۵۹
- شکل (۴-۱) درخت جستجوی BIRA متقارن با دو افزونه‌ی ردیف و دو افزونه‌ی ستون..... ۶۵
- شکل (۴-۲) دیاگرام بلوکی روش پیشنهادی BIRA متقارن توسعه یافته..... ۶۶
- شکل (۴-۳) حافظه‌ی کلمه‌گرای معیوب ۸×۲×۴..... ۶۷
- شکل (۴-۴) استراتژی موفق برای تعمیر حافظه‌ی کلمه‌گرای معیوب شکل (۴-۳) توسط روش پیشنهادی BIRA متقارن توسعه یافته..... ۶۷
- شکل (۴-۵) دیاگرام بلوکی تحلیل‌گر پیشنهادی BIRA متقارن توسعه یافته..... ۶۹
- شکل (۴-۶) نتیجه شبیه‌سازی BIRA متقارن توسعه یافته با استفاده از آدرس‌های خطای شکل (۴-۳)..... ۷۱
- شکل (۴-۷) نتیجه‌ی پیاده‌سازی BIRA متقارن توسعه یافته بر روی تراشه‌ی XC5VLX50T..... ۷۲
- شکل (۴-۸) مقایسه‌ی تعداد سیکل‌های مورد نیاز تحلیل‌گرهای مختلف برای شروع تحلیل بر حسب تعداد خطا..... ۷۸
- شکل (۴-۹) مقایسه‌ی نرخ تعمیر روش پیشنهادی و روش‌های موجود، به ازای آرایش مختلف افزونه‌ها در یک حافظه‌ی ۸۱۹۲×۶۴..... ۷۹
- شکل (پ-۱) حافظه‌ی ۸×۸×۴ معیوب ۱..... ۸۹
- شکل (پ-۲) نتیجه‌ی پیاده‌سازی Wortdy-R-CRESTA بر روی حافظه‌ی شکل (پ-۱)..... ۹۰
- شکل (پ-۳) حافظه‌ی ۸×۸×۴ معیوب ۲..... ۹۱
- شکل (پ-۴) نتیجه‌ی پیاده‌سازی Wortdy-R-CRESTA بر روی حافظه‌ی شکل (پ-۳)..... ۹۲
- شکل (پ-۵) حافظه‌ی ۸×۸×۴ معیوب ۳..... ۹۳
- شکل (پ-۶) نتیجه‌ی پیاده‌سازی Wortdy-R-CRESTA بر روی حافظه‌ی شکل (پ-۵)..... ۹۳
- شکل (پ-۷) حافظه‌ی ۸×۲×۴ معیوب ۱..... ۹۴
- شکل (پ-۸) نتیجه‌ی پیاده‌سازی BIRA متقارن توسعه یافته بر روی حافظه‌ی شکل (پ-۷)..... ۹۵
- شکل (پ-۹) حافظه‌ی ۸×۲×۴ معیوب ۲..... ۹۶

شکل (پ- ۱۰) نتیجه‌ی پیاده‌سازی BIRAی متقارن توسعه یافته بر روی حافظه‌ی شکل (پ- ۹)..... ۹۷

شکل (پ- ۱۱) حافظه‌ی $۸ \times ۲ \times ۴$ معیوب ۳..... ۹۸

شکل (پ- ۱۲) نتیجه‌ی پیاده‌سازی BIRAی متقارن توسعه یافته بر روی حافظه‌ی شکل (پ- ۱۱)..... ۹۹

فهرست حروف اختصاری

ASIC	Application-Specific Integrated Circuit
ATE	Automatic Test Equipment
BIRA	Built-In Redundancy Analyzer
BISD	Built-In Self Diagnosis
BISR	Built-In Self Repair
BIST	Built-In Self-Test
CAM	Content Addressable Memory
CRESTA	Comprehensive Realtime Exhaustive Search Test and Analysis
CRV	Column Repair Vector
DRAM	Dynamic Random Access Memory
DSP	Digital Signal Processing
ECC	Error Checking and Correction
EESP	Extended Essential Spare Pivoting
ESP	Essential Spare Pivoting
FPGA	Field Programmable Gate Array
HDK	Hardware Development Kit
ITRS	International Technology Roadmap for Semiconductors
JTAG	Joint Test Action Group
LRM	Local Repair Most
LSI	Large Scale Integration
MRA	Must-Repair Analyzer
PCI	Peripheral Component Interconnect
R-CRESTA	Reduced Comprehensive Realtime Exhaustive Search Test and Analysis
SDK	Software Development Kit
SOC	System On Chip
SRAM	Static Random Access Memory
VLSI	Very Large Scale Integration

Abstract

Built-In Redundancy Analyzer Synthesizable Design for Word-Oriented Memories

Shahin Khodadadi

With the advance of VLSI circuit's technology and the reduction in the size of devices, the probability of the defects especially in embedded memories has increased. Memory defects cause yield drop and decrease in reliability. Lots of researches have been conducted to design Built-In Self-Repair (BISR) circuits so as to prevent the yield drop get to high reliability. A number of memories are used to save data, since digital circuits highly require saving data. Memories are of two types, based on the cells arrangement: Bit-oriented and word-oriented memories.

ATE and off-chip algorithms were used in former methods. However, using ATE become more and more difficult because of an increase in the complexity of SOC cores. This in turn, has increased the necessity of using Built-In Self-Test (BIST) and Built-In Redundancy Analyzer (BIRA). In contemporary methods, parts of the memory are used as spare redundancies. Furthermore, they are used to substitute the faulty cells of the memory. The most vital component of a BISR is called BIRA which analyzes fault addresses after receiving data from BIST and sends the analysis result to reconfiguration mechanism circuit. In the end, the result of analysis causes the data to change its direction towards the redundancies.

This thesis deals with designing two BIRAs in order to repair word-oriented memories. The first design is proposed to repair multiple-bit errors. In this design the facilitation of achieving optimal repair rate as well as at-speed repair costs using more redundancies. The second design has been offered based on the features of bit-oriented and word-oriented memories as well as the techniques for address transformation in order to repair single-bit errors, which achieves a high repair rate besides at-speed analysis. Not only do the two proposed methods have the above advantages but also they consume a lower area of the chip for implementation purposes. The first proposed design has been improved 78.8% and 63.6% respectively, compared to the EESP and 1-D bitmap methods. The second proposed design also shows a 59.5% and 30.4% improvement in the area consumption.

Keywords:

Embedded Memories, Word-Oriented Memories, Built-In Self-Test, Built-In Self-Repair, Built-In Redundancy Analyzer

مقدمه

۱-۱ مقدمه

چگالی مجتمع‌سازی و عملکرد مدارهای مجتمع طی دو دهه‌ی گذشته تحولات چشم‌گیری را پشت سر گذاشته است. در دهه‌ی ۱۹۶۰، گوردون مور^۱ پیش‌بینی کرد که تعداد ترانزیستورهای یک مدار مجتمع با زمان به طور نمایی رشد می‌کند. بعدها با مشاهده‌ی درستی این پیش‌بینی، آن را قانون مور^۲ نامیدند [۱-۲]. جدول (۱-۱) تعداد ترانزیستورهای موجود در سطح تراشه را طی سال‌های متفاوت نشان می‌دهد که توسط ITRS^۳ ارائه شده است.

جدول (۱-۱) تعداد ترانزیستورهای مجتمع طی سال‌های متفاوت [۱]

سال	تعداد ترانزیستورها
۱۹۷۸	۲۹,۰۰۰
۱۹۸۲	۲۷۵,۰۰۰
۱۹۸۵	۱,۲۰۰,۰۰۰
۱۹۹۱	۳,۱۰۰,۰۰۰
۱۹۹۳	۷,۵۰۰,۰۰۰
۱۹۹۷	۹,۵۰۰,۰۰۰
۲۰۰۱	۵۵,۰۰۰,۰۰۰

پیشرفت تکنولوژی VLSI^۴ سبب استفاده‌ی گسترده این تکنولوژی در سیستم‌های بزرگ دیجیتال مانند ریزپردازنده‌ها^۵، پردازنده‌های سیگنال دیجیتال (DSPs)، حافظه‌های با ظرفیت بالا، کنترل‌کننده‌های I/O و غیره شده است. از طرف دیگر به دلیل اهمیت ذخیره‌سازی داده‌ها در پردازش‌های دیجیتال نیاز به حافظه‌های بزرگ‌تر روز به روز بیشتر می‌شود.

1 Gordon Moore (1929)

2 Moore's law

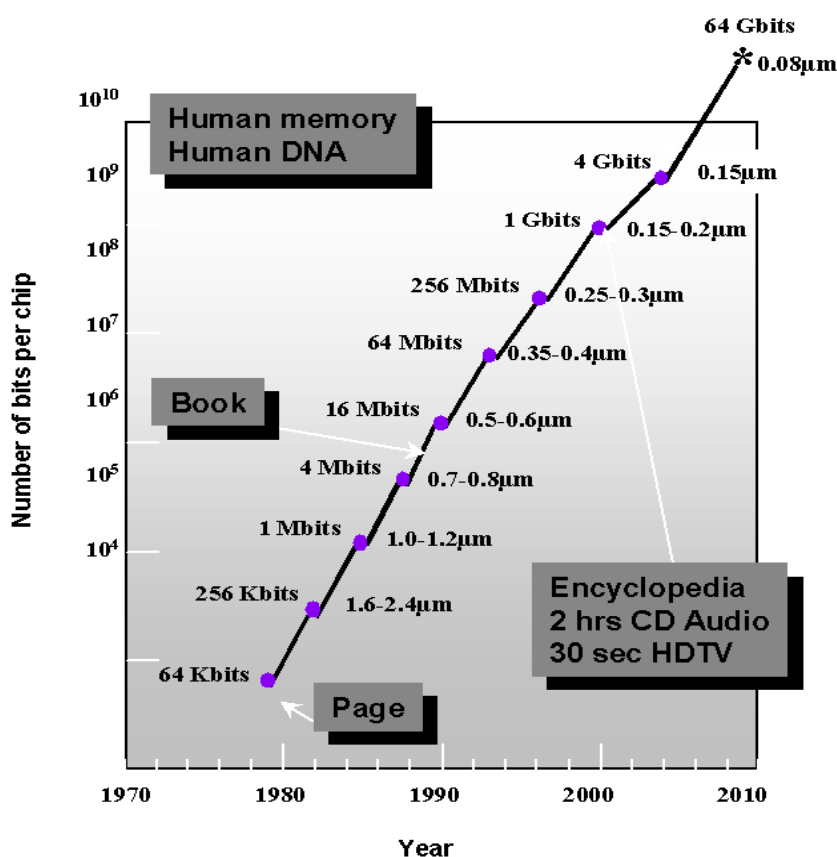
3 International Technology Roadmap for Semiconductors

4 Very Large Scale Integration

5 Microprocessor

6 Digital Signal Processor

شکل (۱-۱) روند افزایش پیچیدگی حافظه را نشان می‌دهد. در این شکل چگالی مجتمع‌سازی حافظه بر حسب زمان مورد بررسی قرار گرفته است. همان طور که مشاهده می‌شود، چگالی حافظه از سال ۱۹۷۰ بیش از ۱۰۰۰ برابر شده است [۳]. هم اکنون در تراشه‌هایی مانند Strong ARM1100 ۹۰ درصد از فضای تراشه به حافظه اختصاص دارد [۴]. با توجه به اینکه بخش قابل توجهی از سطح تراشه را حافظه‌های جاسازی شده^۱ تشکیل داده‌اند، از همین رو بهره‌دهی این حافظه‌ها در میزان بهره‌دهی کل سیستم بسیار موثر خواهد بود.



شکل (۱-۱) روند افزایش پیچیدگی حافظه [۲]

وجود خطا یکی از عواملی است که می‌تواند سبب کاهش میزان بهره‌دهی^۱ در سیستم شود. برای جبران کاهش میزان بهره‌دهی که در اثر وقوع خطا اتفاق روی داده است می‌توان به تعمیر خطا پرداخت. در ابتدا روش‌های تعمیر به گونه‌ای

¹ Embedded Memories

بودند که از تجهیزات آزمون خودکار (ATE^۱) و الگوریتم‌های خارج تراشه استفاده می‌گردید. اما افزایش چگالی مدارات مجتمع، منجر به دشوار شدن ارتباط با I/O، کافی نبودن سرعت و زیاد شدن میزان توان مصرفی مدارات ATE شد که در نتیجه کاهش استفاده از این روش‌ها را به دنبال داشته است [۷-۵]. علاوه بر این روش‌های مبتنی بر ATE تنها خطاهای ناشی از نقص‌های^۳ تراشه را تشخیص می‌دادند و قادر به آشکارسازی خطاهای تصادفی یا خطاهای ناشی از تغییر دما مانند خطاهای نرم نبود. با گسترش تحقیقات در این زمینه، روش‌های آزمون درون‌ساخته (BIST^۴) برای جایگزینی با مدارات ATE ارائه شدند که امروزه از آن‌ها استفاده فراوانی می‌شود. با این روش‌ها تست حافظه بسیار ارزان‌تر از روش‌های سنتی خواهد بود. از طرف دیگر تلاش برای بهبود میزان بهره‌دهی و قابلیت اطمینان حافظه‌های جاسازی شده^۵ منجر به ارائه ایده‌هایی برای خودتعمیری حافظه شد. ایده‌ی تعمیر حافظه با استفاده از افزونه‌ها در دهه‌ی ۱۹۶۰ توسط Tammaru و Angell مطرح شد و طی یک دهه بعد به طور عملی پیاده‌سازی گردید [۸]. هدف اصلی این ایده افزایش میزان بهره‌دهی مدارات LSI^۶ بود. این ایده نیز به عنوان یکی از روش‌های موثر بهبود بهره‌دهی مورد توجه قرار گرفته است. در این روش با در نظر گرفتن افزونه‌های کمکی^۷ برای یک حافظه و طراحی یک مدار تحلیل‌گر افزونه‌ی درون‌ساخته (BIRA^۸) به تعمیر حافظه می‌پردازند. تنها هزینه‌ای که BIST و BIRA در هنگام ساخت حافظه به آن تحمیل می‌کنند، اندکی فضا به منظور پیاده‌سازی آن‌ها می‌باشد. در این روش سلول‌های معیوب با سلول‌های سالم که از پیش در حافظه‌ی اصلی تعبیه شده‌اند جایگزین می‌شوند. عمل جایگزینی با استفاده از یک سیستم برنامه‌پذیر روی تراشه انجام می‌پذیرد. از طرف دیگر با توجه به راحت‌تر بودن نحوه‌ی ساخت حافظه‌های کلمه‌گرا^۹ و راحت‌تر بودن کدنویسی برای این دسته از حافظه‌ها، استفاده‌ی زیادی از آن‌ها می‌شود. در این پایان‌نامه قصد داریم با ارائه یک الگوریتم بهینه برای تعمیر حافظه‌های کلمه‌گرا بپردازیم. در الگوریتم پیشنهادی ضمن استفاده از روش موازی برای بالا نگه داشتن سرعت تحلیل، سطح مصرفی مدار به میزان قابل توجهی کاهش یافته است.

1 Yield

2 Automatic Test Equipment

3 Defects

4 Built-In Self-Test

5 Embedded Memories

6 Large Scale Integration

7 Spare Redundancies

8 Built-In Redundancy Analyzer

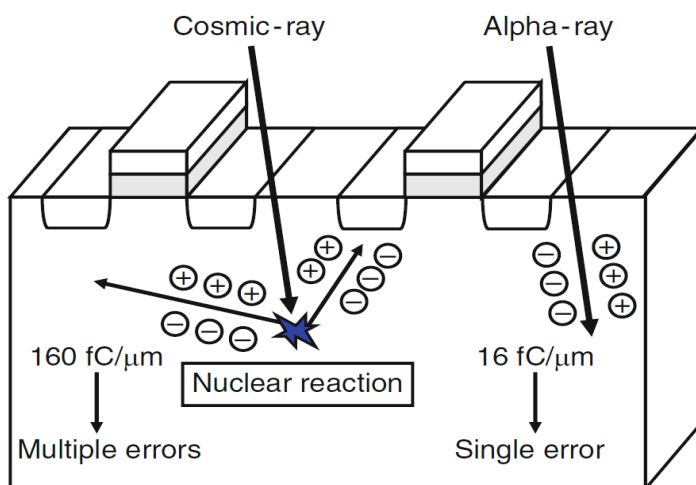
9 Word-Oriented Memories

۱-۲ مفاهیم اولیه

۱-۲-۱ خطای نرم^۱ و خطای سخت^۲

با کاهش چگالی ادوات الکترونیکی در سطوح مجتمع، انواع گوناگونی از نقص‌ها و خطاها پدید می‌آیند. منابع زیادی نیز وجود دارند که می‌توانند منجر به ایجاد خطا در تراشه شوند. خطاهای مربوط به حافظه‌ها را به طور کلی به دو دسته‌ی عمده تقسیم‌بندی می‌کنند که عبارتند از: خطاهای نرم و خطاهای سخت.

خطای نرم موجب از بین رفتن اطلاعات ذخیره شده در سلول‌های حافظه و دیگر گره‌های مدار می‌شود. این خطاها اغلب به دلیل نویز به خصوص ذرات آلفا و یا پرتوهای کیهانی پدید می‌آیند. خطای نرم و علل آن در شکل (۱-۲) نشان داده شده است. اگر بار تولید شده، بر روی گره‌ای که اطلاعات را ذخیره کرده است، جمع شود می‌تواند سبب شکست در مقدار ولتاژ آن گره شود. تا زمانی که قطعه در اثر خطای نرم تخریب نشده باشد، عملکرد صحیح دستگاه در اثر بازنویسی اطلاعات از دست رفته، امکان‌پذیر خواهد بود [۱۰-۱۱].



شکل (۱-۲) خطای نرم و عوامل آن [۹]

1 Soft error
2 Hard error