
بسم الله الرحمن الرحيم

دانشگاه گیلان

دانشکده فنی

پایان‌نامه کارشناسی ارشد

پیاده‌سازی معماری خودتعمیر برای حافظه‌های جاسازی شده بر اساس استاندارد آزمون هسته‌ها

از:

پیام حبیبی

استاد راهنما:

دکتر راهبه نیارکی اصلی

اسفند ۱۳۹۰

دانشکده فنی

گروه مهندسی برق

گرایش الکترونیک

پیاده‌سازی معماری خودتعمیر برای حافظه‌های جاسازی شده بر اساس استاندارد آزمون هسته‌ها

از:

پیام حبیبی

استادراهنما:

دکتر راهبه نیارکی اصلی

اسفند ۱۳۹۰

تقدیم به:

پدر و مادر عزیز و همسر مهربانم

تشکر و قدردانی

از استاد راهنمای ارجمندم سرکار خانم دکتر راهبه نیارکی اصلی به خاطر زحمات فراوان و رهنمودهای ارزنده-شان در انجام این پایان‌نامه و در تمام طول دوران تحصیل کمال تشکر و امتنان را دارم. حضور ایشان همواره روشنی‌بخش مسیر تحقیق در این پژوهش بوده است.

از خانواده‌ام به‌ویژه پدر و مادر و برادر عزیزم که در تمام مراحل زندگی همواره مشوق و پشتیبان من بودند سپاسگذارم و همچنین از همسر صبور و دلسوزم که در طول این پروژه زحمات و سختی‌های فراوانی را متحمل شدند بسیار متشکرم.

پیام حبیبی

اسفند ۱۳۹۰

چکیده

پیاده‌سازی معماری خودتعمیر برای حافظه‌های جاسازی شده بر اساس استاندارد آزمون هسته‌ها

پیام حبیبی

بیشترین تعداد هسته‌های موجود در سیستم‌های روی تراشه (SOC) را حافظه‌های جاسازی شده تشکیل می‌دهند. از این رو بهره‌دهی این حافظه‌ها نقش بسزایی در بهره‌دهی کل SOC دارد. پیشرفت مداوم مجتمع‌سازی در سطح تراشه‌ها و در نتیجه افزایش چگالی حافظه‌های جاسازی‌شده، امکان بروز نقص در سلول‌های حافظه را افزایش داده است که این امر به کاهش بهره‌دهی کل تراشه منجر می‌گردد. با تعمیر حافظه‌های جاسازی‌شده می‌توان این مشکل را تا حد زیادی برطرف نمود.در صورتی که نقص پدید آمده در حافظه از نوع خطاهای سخت باشد می‌توان با جایگزین کردن سلول‌های معیوب با سلول‌های سالم که به‌صورت افزونه در کنار حافظه قرار می‌گیرند عملیات تعمیر را انجام داد.

روش‌های قدیمی تعمیر، بر استفاده از ابزارهای آزمون خودکار (ATE) و الگوریتم‌های خارج از تراشه استوار بودند. اما امروزه تعداد و پیچیدگی هسته‌های SOCها افزایش یافته و به علت تعداد زیاد I/Oاهای هسته‌های حافظه، دسترسی به همه‌ی آن‌ها از طریق I/Oاهای SOC دشوارتر شده است. به منظور حل این مشکل‌ها که مشاهده‌پذیری و کنترل‌پذیری نامیده می‌شوند و همچنین به علت نیاز به انجام آزمون و تعمیر با سرعت، روش‌های گوناگون خودآزمون درون‌ساخته (BIST) و خود-تعمیری درون‌ساخته (BISR) به حافظه‌های جاسازی‌شده اعمال شدند. مدار BIST بر اساس استاندارد آزمون هسته‌ها و با استفاده از الگوریتم‌های ویژه ، حافظه را می‌آزماید و آدرس محل‌های وقوع خطا را آشکار می‌سازد. تحلیلگر افزونه‌ی درون-ساخته (BIRA) که یکی از عناصر کلیدی BISR است، اطلاعات خطای ارسال شده از سوی BIST را دریافت نموده و بر اساس یک الگوریتم تحلیل، چگونگی تخصیص افزونه‌ها را تعیین می‌کند. پس از این مرحله یک پروسه‌ی پیکربندی مجدد برای غیرفعال‌سازی سلول‌های معیوب حافظه انجام می‌گیرد.

الگوریتم‌های مختلف BIRA با سه چالش مهم روبرو هستند: سطح اشغالی تراشه، نرخ تعمیر و سرعت تحلیل. در این پایان‌نامه یک مدار BIRAی جدید با نرخ تعمیر بهینه با استفاده از زیرتحلیلگرهای موازی طراحی و پیاده‌سازی شده است. یکی از مشکلات اساسی روش‌های تحلیل موازی موجود این است که فضای زیادی را از سطح تراشه اشغال می‌کنند. در روش پیشنهادی تلاش شده است که این ضعف بهبود یابد. مدار طراحی شده نسبت به تحلیلگر موازی R-CRESTA ۵۰ درصد فضای کمتری از سطح تراشه اشغال می‌نماید. این برتری با پذیرش هزینه‌ی حداکثر یک تکرار آزمون به‌دست آمده است. با این وجود تحلیلگر پیشنهادی همچنان نسبت به تحلیلگرهای ESP و IntelligentSolveFirst از سرعت بالاتری برخوردار است.

واژه‌های کلیدی:

Embedded Memory, SOC, Built-in Self-Test, Built-in Self-Repair, Built-in Redundancy Analyzer

فهرست مطالب

فهرست شکل‌ها

فهرست جدول‌ها

فهرست علائم اختصاری

ADC ..... Analog to Digital Converter
ASIC ..... Application-Specific Integrated Circuit
ATE ..... Automatic Test Equipment
BIRA ..... Built-in Redundancy Analayzer
BISR ..... Built-in Self Repair
BIST ..... Built-in Self Test
CAM ..... Content Addressable Memory
CMOS ..... Complementary Metal-Oxide Semiconductor
CPLD ..... Complex Programmable Logic Device
CRESTA ..... Comprehensive Realtime Exhaustive Search Test and Analysis
DAC ..... Digital to Analog Converter
DBL ..... Divided Bit Line
DFT ..... Design for Test
DRAM ..... Dynamic Random Access Memory
DWL ..... Divided Word Line
ECC ..... Error Checking and Correction
ELRM ..... Extended Local Repair Most
EOF ..... Early termination by the number of Orthogonal Faulty cells
ESP ..... Essential Spare Pivoting
FPGA ..... Field Programmable Gate Array
ITRS ..... International Technology Roadmap for Semiconductors
LO ..... Local Optimization
LRM ..... Local Repair Most
NP ..... Nondeterministic Polynomial time
NV memory ..... Non Volatile memory
R-CRESTA ..... Reduced Comprehensive Realtime Exhaustive Search Test and Analysis
RISC ..... Reduced Instruction Set Computer
RM ..... Repair Most
SOC ..... System On Chip
SRAM ..... Static Random Access Memory
TAM ..... Test Access Mechanism
ULSI ..... Ultra Large Scale Integration
VLSI ..... Very Large Scale Integration

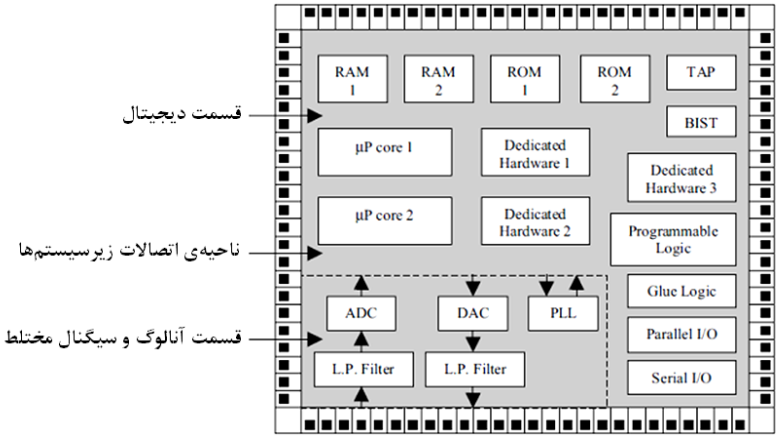
#

فصل اول:

مقدمه##

۱-۱- مقدمه

با پیشرفت فن‌آوری CMOS[1] در قلمرو زیرمیکرون، میزان مجتمع‌سازی، کارایی و هزینه‌ی ساخت به‌طور چشمگیری افزایش می‌یابد. به‌همین دلیل در ساخت مدارهای کوچک که نیاز به مجتمع‌سازی بالا ندارند،استفاده از این فن‌آوری مقرون‌به‌صرفه نیست و تنها تراشه‌های سیستمی با کارایی بالاارزش پیاده‌سازی با این روش را دارند. به این تراشه‌های سیستمی، سیستم‌روی‌تراشه[2] گفته می‌شود[۱]. نمونه‌ای از یک سیستم‌روی‌تراشه در شکل (۱-۱) دیده می‌شود که شامل هسته‌های پردازنده، حافظه‌های جاسازی‌شده، DAC[3] و ADC[4] و غیره می‌باشد.

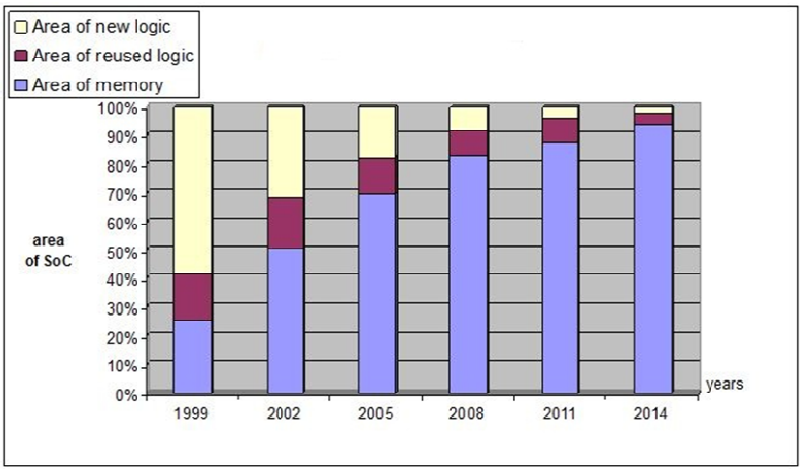


شکل (۱-۱) مثالی از یک سیستم روی تراشه [۲]

دو علت برای تبدیل حافظه‌های جاسازی شده به یک عنصر کلیدی در SOCها وجود دارد.دلیل اول این است که فن‌آوری CMOSامکانات مناسبی جهت مجتمع‌سازی حافظه‌های بزرگ با مدارات دیگر ارائه می‌دهد. این فن-آوری قابلیت پیاده‌سازی تراشه‌های ULSI[5] با بیش از $۱۰^۹$ عنصر روی یک تراشه را دارا می‌باشد. پردازش داده‌ها و ذخیره‌سازی آن‌ها از اساسی‌ترین مولفه‌های مدارهای دیجیتال هستند. بنابراین پیاده‌سازی حافظه‌ها روی تراشه از اولویت بالایی برخوردار است.دلیل دیگر اینست که پهنای باند حافظه یکی از چالش‌های جدی در عملکرد سیستم-ها است. محدودیت سرعت I/Oها باعث ایجاد شکاف بین سرعت ریزپردازنده‌ها و حافظه‌ها می‌شود. این شکاف سرعت باعث افزایش تقاضا برای مجتمع‌سازی حافظه‌ها با ریزپردازنده‌ها روی یک تراشه شده است.

[1] Complementary Metal-Oxide Semiconductor
[2] System On Chip
[3] Digital to Analog Converter
[4] Analog to Digital Converter
[5] Ultra Large Scale Integration

معماری RISC[1] شدیداً وابسته به پهنای باند حافظه است به‌گونه‌ای که ریزپردازنده‌های RISC با کارایی بالا، دارای فرکانس بیش از ۲۵ تا ۵۰ مگاهرتز، با حافظه‌ی Cache روی تراشه تجهیز می‌شوند[۲]. بر اساس پیش‌بینی ITRS 2001[2] [۳]، تا سال ۲۰۱۴ بیش از ۹۰ درصد فضای اشغالی SOCها را حافظه‌ها به خود اختصاص خواهند داد [۷-۴]. شکل (۱-۲) نموداری از رشد مساحت حافظه‌ها نسبت به مساحت SOC را نشان می‌دهد. هم‌اکنون در تراشه‌هایی مانند Strong ARM1100 ۹۰درصد از فضای تراشه به حافظه اختصاص یافته است [۸]. همچنین ۱۳۵میلیون ترانزیستور از ۱۵۲میلیون ترانزیستور تراشه‌ی Compaq Alpha EV7 مربوط به حافظه‌هایتراشه است.



شکل (۱-۲) نمودار رشد فضای اشغالی حافظه نسبت به کل فضای SOC[۳]

با کاهش طول گیت به زیر ۲۰ نانومتر در سیستم‌های VLSI[3] چالش‌های جدیدی در زمینه‌ی طراحی پدید می‌آید. با پیچیده‌تر شدن پروسه‌ی ساخت، نقص‌های حین ساخت افزایش می‌یابند و این درحالی است که به علت پیچیده‌تر شدن سیستم، آزمودن آن بسیار دشوارتر می‌گردد. مشکلاتی مانند نویز هم‌شنوایی[4]، افت ولتاژ، اتلاف توان، توزیع گرمایی سبب ایجاد انواع خطاها در حافظه و کاهش قابلیت اطمینان[5] می‌شود [۹]. علاوه‌براین کوچک شدن اندازه مشخصه[6]، تراشه را مستعد نقص‌های غیرقابل پیش‌بینی می‌نماید که ناشی از ذرات آلفا بوده و خطای

[1] Reduced Instruction Set Computer
[2] International Technology Roadmap for Semiconductors
[3] Very Large Scale Integration
[4] Cross talk
[5] Reliability
[6] Feature size

نرم[1] نامیده می‌شوند.

پیشرفت‌های زیادی در تکنیک‌هایی همچون طراحی آزمون‌پذیر[2] و خودآزمون درون‌ساخته[3] صورت گرفته است. روش‌های آزمون در گذشته بر استفاده از ابزارهای خارج از تراشه یا ATE[4] استوار بودند [۱۳-۱۰].اما در مدارهای مجتمع کنونی کارآیی خود را از دست داده‌اند.زیرا روش‌های مبتنی بر ATEتنها خطاهای ناشی از نقص‌های تراشه[5] را تشخیص می‌دادند اما قادر به آشکارسازی خطاهای تصادفی یا خطاهای ناشی از تغییر دما مثل خطاهای نرم نبودند [۱۴]. از سوی دیگر در یک SOC حافظه‌هایی با اندازه‌ها و معماری‌های گوناگون به‌کار رفته است و باید از ابزارهای آزمون و تعمیر با پیکربندی‌های متفاوت استفاده شود که این مسئله خود منجر به تحمیل هزینه‌های مضاعفمی‌شود. با افزایش مجتمع‌سازی SOCها امکان دسترسی به پایه‌های حافظه‌ی تحت آزمون کاهش می‌یابد. بنابراین به مرور زمان برای رفع این مشکلات، کاهش هزینه، افزایش سرعت و افزودن قابلیت تعمیر با سرعت[6]، روش‌های خارج از تراشه جای خود را به رهیافت‌های درون‌ساخته و روی تراشه دادند.

وقوع نقص یا خطا در سلول‌های حافظه، به کاهش بهره‌دهی[7] آن منجر می‌شود. با توجه به این‌که حافظه بخش عمده‌ی SOC را تشکیل می‌دهد، کاهش بهره‌دهی آن سبب کاهش بهره‌دهی کل مدار می‌شود. جهت رفع این مشکل و همچنین برای دستیابی به قابلیت اطمینان[8] بالا، سیستم باید به‌صورت دینامیک دارای قابلیت تحمل خطا باشد تا بتواند در مرحله‌ی کارکرد، خطاها را رفع کند.یک روش معمول برای حل این مشکل، جایگزین کردن سلول‌های معیوب با سلول‌های سالم یا به اصطلاح افزونه‌ها[9] می‌باشد. این کار توسط مدار خودتعمیر درون‌ساخته[10] که روی تراشه در کنار حافظه قرار می‌گیرد، انجام می‌شود.در[۱۵] با استفاده از این روش ۱۱/۵۳٪ بهبود بهره‌دهی به‌دست آمده است.

[1] Soft error
[2] Design for Test (DFT)
[3] Built-in Self-Test (BIST)
[4] Automatic Test Equipment
[5] Defects
[6] at-speed
[7] Yield
[8] Reliability
[9] Redundancy
[10] Built-in Self-Repair

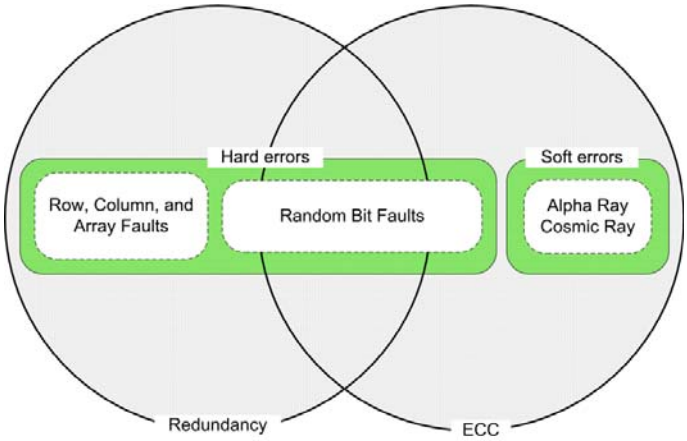
۱-۲- مفاهیم اولیه

۱-۲-۱- انواع خطاها

به دنبال افزایش ظرفیت، کاهش اندازه مشخصه و کاهش ولتاژ کاری حافظه‌های جاسازی شده انواع گوناگونی از نقص‌ها و خطاها پدید می‌آیند. از جمله این خطاها می‌توان خطاهای سخت/نرم، خطاهای حاشیه ولتاژ/زمان و خطاهای مرتبط با سرعت[1] را نام برد. روش‌های اصلاح این خطاها با توجه به نوع خطا متفاوت است. طرح ارائه شده در این پروژه برای تعمیر خطاهای سخت مورد استفاده قرار می‌گیرد. خطاهای سخت که اساساً در اثر نقص-های طی پروسه‌ی ساخت پدید می‌آیند سبب عملکرد نادرست قسمتی از یک تراشه می‌شوند. شکل (۱-۳) انواع خطاهای سخت و روش اصلاح آن‌ها را در کنار خطاهای نرم نشان می‌دهد. خطاهای سخت را می‌توان بر اساس تعداد سلول‌های معیوب و نوع توزیع آن‌ها به سه دسته تقسیم کرد که عبارتند از خطاهای تک بیتی (معیوب بودن تنها یک سلول)، خطاهای ردیف/ستون (معیوب بودن تمام سلول‌های یک ردیف یا ستون) و خطاهای آرایه‌ای (معیوب بودن تمام سلول‌های یک آرایه).

خطاهای نرم خطاهایی هستند که در اثر آن‌ها اطلاعات ذخیره شده در گره‌های سلول‌های حافظه و یا سایر گره‌های مدار از دست می‌رود. این خطاها در اثر نویز به خصوص ذرات آلفا و یا پرتوهای کیهانی پدید می‌آیند. از آن‌جا که در اثر خطاهای نرم حافظه دچار خرابی نمی‌شود، با نوشتن مجدد اطلاعات این مشکل برطرف می‌شود. جهت اصلاح خطاهای نرم از روشی موسوم به ECC[2] استفاده می‌شود [۱۹-۱۶]. به این ترتیب که طی مرحله‌ی نوشتن در حافظه، مدار کدگذاری از روی بیت‌های داده‌ی ورودی چک‌بیت‌هایی را تولید کرده و با استفاده از آن به آشکارسازی و رفع خطاها می‌پردازد. هرچند ECC برای اصلاح هر دو نوع خطا استفاده می‌شود اما اغلب برای تعمیر خطاهای نرم به کار گرفته می‌شود و افزونه‌ها به‌عنوان راه‌حل اصلی تعمیر خطاهای سخت به‌شمار می‌روند [۲۰] که در این پروژه نیز به آن پرداخته شده است.

[1] Speed relevant
[2] Error Checking and Correction



شکل (۱-۳) خطاهای سخت و نرم و راه‌حل اصلاح آن [۲۰]

۱-۲-۲- تحمل خطا[1]

به قابلیت یک سیستم در تداوم اجرای عملیات مورد نظر علی‌رغم بروز نقص، تحمل خطا گفته می‌شود. داشتن چنین ویژگی در کاربردهای حساس مثل موارد پزشکی و هوافضا از اهمیت فراوانی برخوردار می‌باشد. سیستم با تحمل خطا قادر است به روش سخت‌افزاری یا نرم‌افزاری خطاهای غیرقابل پیش‌بینی را برطرف سازد. سیستمی با این ویژگی دارای قابلیت اطمینان بالاتری نیز خواهد بود. اگر قابلیت اطمینان یک عنصر سیستم ٪۹۹/۹۹باشد، قابلیت اطمینان کل سیستم متشکل از ۱۰۰۰۰ قطعه از این عنصر تنها ٪۳۶/۷۹ خواهد بود [۲۱]. بنابراین طراحی کلیه‌ی اجزای سیستم به گونه‌ای که دارای تحمل خطا باشند به بهبود قابلیت اطمینان کل سیستم کمک بسزایی می‌نماید. از آن‌جا که حافظه‌های جاسازی شده عنصر غالب در SOCها می‌باشند، در این پروژه تلاش شده است که با افزودن ویژگی خودتعمیری، تحمل خطا در حافظه افزایش یافته و در نتیجه قابلیت اطمینان آن نیز بهبود یابد.

با وجود تلاش طراحان و سازندگان در ارائه‌ی محصول بی‌نقص، بروز خطا در سیستم اجتناب‌ناپذیر است. عامل برخی از این خطاها عوامل محیطی بوده که از کنترل سازنده خارج است. راه‌های گوناگونی جهت دستیابی به یک سیستم دارای تحمل خطا وجود دارد که یکی از موارد رایج آن استفاده از عناصر جایگزین یا افزونه‌ها می‌باشد. افزونه‌ها به دو دسته‌ی اصلی افزونه‌یفضایی و افزونه‌ی زمانی تقسیم می‌شوند. در جایگزینی فضایی عناصر یا داده-هایی مضاعف به سیستم افزوده می‌شود که در صورت عدم وقوع خطا بدون استفاده خواهند ماند. این نوع

[1] Fault tolerance

افزونهخود به سه دسته‌ی سخت‌افزاری، نرم‌افزاری و اطلاعاتی تقسیم می‌گردد. در جایگزینی زمـانی، محاسـبات یـا داده‌های ارسالی تکرار می‌شوند و نتیجه‌ی حاصل با نتیجه‌ی ذخیره شده‌ی پیشین مقایسه می‌گردد.

منظور از افزونه در این پایان‌نامه جایگزین فضایی نوع سخت‌افزاری می‌باشد. حافظه از آرایـه‌ای از سـلول‌هـای حافظه تشکیل می‌شود که در ردیف‌ها و ستون‌هایی کنار یکدیگر قرار گرفته‌اند. بنابراین می‌توان جایگزین‌هـایی بـه صورت سلولی، ردیفی، ستونی و بلوکی را برای آن در نظر گرفت. جهـت انتخـاب یکـی از ایـن جـایگزین‌هـا بـرای حافظه‌ی خودتعمیر، دو سیستم شکل (۱-۴) را در نظر می‌گیریم.

(ب) (الف)

شکل (۱-۴) الف) جایگزینی سطح بالا و ب) سطح پایین

در شکل (۱-۴-الف) که نمونه‌ای از جایگزینی سطح بالا می‌باشد کل سیستم متشکل از اجـزای ۱ و۲ و ۳ بـه-عنوان جایگزین در کنار سیستم اصلی قرار گرفته است در حالی که در شکل(۱-۴-ب)که نشان‌دهنده‌ی جـایگزینی سطح پایین می‌باشد به ازای هر یک از اجزای سیستم یک جایگزین قـرار داده شـده اسـت. اگـر قابلیـت اطمینـان سیستم‌های الف و ب را به ترتیب با R_a و R_b نشان دهیم مقادیر زیر را خواهیم داشت [۲۱]:

$$R_a = 1 - (1 - R_1R_2R_3)^2 \qquad (۱-۱)$$

$$R_b = \left(1 - (1 - R_1)^2\right)\left(1 - (1 - R_2)^2\right)\left(1 - (1 - R_3)^2\right) \qquad (۱-۲)$$

برای مقایسه‌ی قابلیت اطمینان دو سیستم این دو مقدار را از هم کم می‌کنیم. با فرض این‌که قابلیت اطمینـان همه‌ی اجزا برابر با R باشد خواهیم داشت:

$$R_b - R_a = 6R^3(1 - R)^2 \qquad (۱-۳)$$

بنابراین R_b همواره از R_a بزرگتر خواهد بود یعنی در صورتی که جایگزینی را در سطح عناصر کوچکتر سیسـتم انجام دهیم به قابلیت اطمینان بالاتری دست خواهیم یافت.

۱-۲-۳- بهره‌دهی

بهره‌دهی یک تراشه به صورت نسبت اجزای سالم و قابل قبول به کل اجزای ساخته شده در پروسه ساخت تعریف می‌شود.یکی از مشکلاتی که با رشد مجتمع‌سازی حافظه‌های جداسازی شده پدید می‌آید، کاهش بهره‌دهی حافظه و در نتیجه کاهش بهره‌دهی کل سیستم است. افزونه‌ها به‌طور گسترده به‌عنوان یک راه‌حل موثر در بهبود بهره‌دهی و کاهش هزینه به‌کار گرفته شده‌اند. برای تحقیق این مسئله ابتدا باید مدلی از توزیع خطا در دست داشته باشیم.

یک تراشه حافظه با N عنصر را در نظر می‌گیریم، این عناصر می‌توانند یک سلول، ردیف، ستون و یا حتی یک آرایه حافظه باشند. اگر احتمال وقوع نقص در یک عنصر p باشد و حافظه دارای N عنصر باشد، آنگاه احتمال معیوب بودن k عنصر از حافظه را می‌توان با توزیع دوجمله‌ای زیر نمایش داد [۲۰]:

$$p(k) = \binom{N}{k} p^k (1-p)^{N-k} \qquad (۴-۱)$$

اغلب N بسیار بزرگ و p بسیار کوچک است، با فرض $\lambda = Np$ خواهیم داشت:

$$p(k) = \frac{N(N-1)\cdots(N-k+1)}{K!} p^k p^{N-k} \qquad (۵-۱)$$

$$= 1.\left(1-\frac{1}{N}\right)\cdots\left(1-\frac{k+1}{N}\right)\frac{\lambda^k}{k!}(1-\frac{\lambda}{N})^N(1-\frac{\lambda}{N})^{-k}$$

اگر $N \longrightarrow \infty$ به عبارت (۱-۶) می‌رسیم که یک توزیع پوآسن است:

$$p(k) = \frac{\lambda^k}{k!}(1-\frac{\lambda}{N})^N = \frac{\lambda^k e^{-\lambda}}{k!} \qquad (۶-۱)$$

مدل پواسن به علت سادگی ریاضی، برای محاسبه بهره‌دهی حافظه مورد استفاده قرار می‌گیرد [۲۳، ۲۲، ۱۱]. ثابت می‌شود که λ برابر با میانگین تعداد خطاها می‌باشد. بنابراین می‌توان آن را به‌صورت زیر نشان داد:

$$\lambda = AD \qquad (۷-۱)$$

در این رابطه A مساحت تراشه و D چگالی خطا می‌باشد. احتمال بی‌خطا بودن یک تراشه با قرار دادن $k = 0$ به‌دست می‌آید و این مقدار با بهره‌دهی حافظه بدون جایگزین برابر است: