بهبود قابلیت تشخیص خطاهای مجاور ناشی ازتشعشعات در FPGA های مورد استفاده در محیط‌های خشن

احمد رحمانفر[1]، یاسر بالغی[2]، محمد رضا ذهابی [3]

[1]کارشناسی ارشد برق الکترونیک، دانشگاه صنعتی نوشیروانی بابل، rahmanfar.ahmad@gmail.com

[2]استادیار دانشکده برق و کامپیوتر، دانشگاه صنعتی نوشیروانی بابل

[3]استادیار دانشکده برق و کامپیوتر، دانشگاه صنعتی نوشیروانی بابل

تاریخ دریافت :۱۳۹۴/۰۷/۲۰ تاریخ پذیرش :۱۳۹۵/۰۸/۰۲

چکیده

استفاده از FPGA های مبتنی بر SRAM در کاربردهای محیط‌های خشن مانند عملیات صنعتی، نظامی و یا اکتشاف دور به دلیل قابلیت باز پیکربندی در این قطعات بسیار مورد توجه بوده است. نتایج آزمون‌های به دست آمده بر روی FPGA های مبتنی بر SRAM نشان می‌دهد که این قطعات فوق العاده به تشعشعات حساس هستند و نرخ SEU در آنها بسیار زیاد است. کد همینگ برای مقابله با SEU در بیت‌های پیکره بندی ماژول سویچ FPGA های مبتنی بر SRAM استفاده شده است. این کد قابلیت تصحیح خطاهای تک بیتی را دارد، اما با پیشرفت تکنولوژی قطعات نیمه هادی و افزایش چگالی حافظه‌ها، یک ذره پر انرژی از تشعشعات فضایی می تواند چند بیت حافظه را به صورت هم زمان دچار SEU گرداند که در اکثر موارد این بیت‌ها مجاور هستند. در این مقاله دو جایابی بیت جدید برای بیت‌های پیکره بندی ماژول سویچ ارائه شده است که موجب افزایش قابلیت تشخیص خطاهای مجاور دوتائی از ۱۱/۱۱٪ به ۶۶/۶۶٪ (در مقاوم سازی با کد همینگ در سطح جعبه سویچ) و از ۳/۷۵٪ به ۷۵٪ (در مقاوم سازی با کد همینگ در سطح ماژول سویچ) می‌شوند. همچنین یک روش دیگر مبتنی بر الگوریتم ژنتیک ارائه شده است که اعمال آن (در مقاوم سازی با کد همینگ در سطح جعبه سویچ) موجب تشخیص ۸۸/۸۸٪ خطاهای مجاور دوتائی می‌شود. هر دو روش اعمال شده (جایابی بیت انتخابی و روش ارائه شده مبتنی بر الگوریتم ژنتیک) هیچگونه افزونگی را تحمیل نمی کنند.

کلید واژه

FPGA های مبتنی بر SRAM، SEU، جعبه سویچ، ماژول سویچ، کد همینگ.

مقدمه

در محیط‌های خشن مانند محیط‌های صنعتی، نظامی و یا فضایی تجهیزات الکترونیکی از تشعشعاتِ مختلف در امان نیستند. یکی از اثرات مخرب این تشعشعات در FPGA[1] های مبتنی بر SRAM[2]، SEU[3] است، که عبارت است از تغییر حالت (از صفر به یک و بر عکس) بیت‌های حافظه [۱]. نتایج آزمون‌های به دست آمده بر روی FPGAهای مبتنی بر SRAM نشان می‌دهد که این قطعات فوق العاده به تشعشعات در محیط‌های خشن حساس هستند[۲]. اما استفاده از آنها در کاربردهایی مانند عملیات صنعتی، نظامی و یا اکتشاف دور به دلیل قابلیت باز پیکربندی که در این قطعات قرار داده شده است بسیار ضروری است[۳]. بیت‌های پیکره بندی در FPGA برای مشخص کردن توابع مدار و چگونگی اتصالات آنها استفاده می‌شوند. SEU در بیت‌های پیکره‌بندی موجب تغییر تابع اجرایی مدار و تغییر اتصالات و در نهایت خرابی سیستم مورد نظر می‌شود[۴].

در سال‌های اخیر با پیشرفت تکنولوژی قطعات نیمه‌هادی و افزایش چگالی حافظه‌ها، یک ذره پر انرژی از تشعشعات محیطی می‌تواند چند بیت حافظه را به صورت هم زمان دچار SEU گرداند [۶،۵]. این بیت‌های واژگون شده به هم نزدیک ودر اکثر موارد مجاور هستند [۷]. در نتیجه تشخیص وتصحیح خطا در بیت‌های مجاور اهمیت می‌یابد.

روش‌های کلاسیک متعددی برای مقابله با SEU وجود دارد، که به دو دسته روش‌های افزونگی سخت افزاری و روش-های افزونگی اطلاعات تقسیم می‌شوند. به عنوان مثال، از روش‌های سخت افزاری می‌توان به افزونگی سه ماژوله اشاره کرد[۸]. در این روش هر ماژول سه بار تکرار شده و خروجی نهایی از رای گیری بین ماژول ها استخراج می‌شود. زمانی که SEU موجب خرابی یکی از ماژول‌ها شود، مدار رای گیر مقدار صحیح را با استفاده از دو ماژول دیگر عبور می‌دهد. این روش درصد سرباری بیشتر از دو برابر را برای توان و مساحت مدار به

[1] Field Programmable Gate Array
[2] Static Random Access Memory
[3] Single Event Upset

همراه دارد و اگر دو ماژول به طور هم زمان دچار SEU شوند، مدار رای گیر مقدار غلط را به خروجی عبور خواهد داد.

از روش‌های افزونگی اطلاعات نیز می‌توان کد دو بعدی [۹]، کد رید سالومون [۱۰] و کدهای همینگ [۱۱] را نام برد. کد دو بعدی دارای قابلیت تصحیح خطای بالایی است، اما این کد درصد سرباری زیادی برای توان و مساحت مدار به همراه دارد. کد رید سالومون نیز می‌تواند برای مقابله با خطاهای چند بیتی استفاده شود، اما این کد به دلیل داشتن الگوریتم کدگذاری و کد برداری پیچیده، درصد سر باری زیادی را در مساحت، توان و زمان به مدار تحمیل می‌کند. کد همینگ با قابلیت تصحیح خطای تک بیتی، برای مقابله با SEU در ماژول سویچ FPGA های مبتنی بر SRAM استفاده شده است[۱۱]. اگر یک خطای تک بیتی در بیت‌های کد شده اتفاق بیافتد بردار سندرم که از حاصلضرب عبارت کد شده با ماتریس H به دست می‌آید، یک توصیف باینری از مکان خطای ایجاد شده در کد را نشان می‌دهد. اما این کد توانایی تشخیص بین خطا در یک بیت و یا خطا در دو بیت را ندارد و در اکثر موارد، کد به جای تشخیص خطای دو بیتی از تک بیتی، به اشتباه یک بیت دیگر را تصحیح می‌کند.

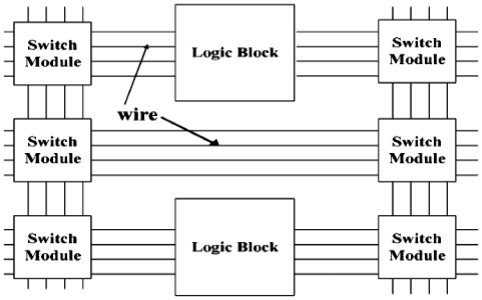
در این مقاله دو جایابی بیت جدید برای بیت‌های پیکره بندی ماژول سویچ FPGA های مبتنی بر SRAM ارائه شده است که موجب افزایش قابلیت تشخیص خطاهای مجاور دوتائی (در مقاوم سازی با کد همینگ در سطح جعبه سویچ و در سطح ماژول سویچ) می‌شوند. همچنین یک روش دیگر مبتنی بر الگوریتم ژنتیک ارائه شده است که اِعمال آن (در مقاوم سازی با کد همینگ در سطح جعبه سویچ) موجب افزایش بیشتر تشخیص خطاهای مجاور دوتائی می‌شود. هر دو روش اعمال شده (جایابی بیت انتخابی و روش ارائه شده مبتنی بر الگوریتم ژنتیک) افزونگی خاصی را تحمیل نمی‌کنند.

در ادامه این مقاله به مروری برکار قبلی پرداخته خواهد شد، سپس استراتژی جایابی بیت انتخابی توضیح داده می‌شود، بعد از آن روش پیشنهادی مبتنی بر الگوریتم ژنتیک معرفی می‌شود، در ادامه نیز به ارائه نتایج پرداخته شده است (که شامل نتایج به دست آمده از دو جایابی بیت پیشنهادی و نتیجه حاصل از اعمال الگوریتم ژنتیک است) و در نهایت نتیجه گیری ارائه شده است.

مروری کلی بر کار قبلی

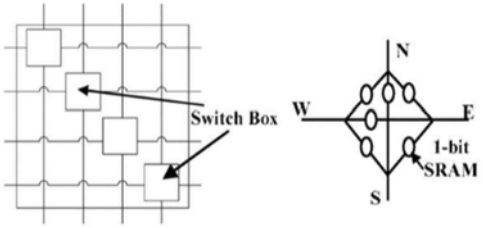
دو نوع حافظه پیکره‌بندی در FPGA ها وجود دارد، یکی برای بلوک‌های منطقی و دیگری برای ماژول‌های سویچ. بیت‌های پیکره‌بندی بلوک‌های منطقی برای ذخیره کردن جداول صحت مربوط به توابع مدار استفاده می‌شوند و بیت‌های پیکره‌بندی مربوط به ماژول‌های سویچ برای تعیین چگونگی اتصالات بین توابع و پایه‌های خروجی مورد استفاده قرار می‌گیرند. معماری این مدل در شکل۱ نشان داده شده است.

همچنین در شکل۲ (الف) یک ماژول سویچ نشان داده شده است (هر ماژول سویچ شامل چهار جعبه سویچ است). همانطور که در شکل۲(ب) نیز نشان داده شده است هر جعبه سویچ شامل چهار سیم ورودی / خروجی است که می‌توانند به سیم-های دیگر متصل شوند. بیت‌های SRAM اتصالات مختلف میان سیم‌ها را کنترل می‌کنند. بر اساس اینکه چه مقداری در هر سلول SRAM ذخیره شده باشد (صفر یا یک)، سیم بندی یک FPGA مبتنی بر SRAM توصیف می‌شود. همانطور که در مقدمه نیز بیان شد SEU در این بیت‌ها موجب تغییر اتصالات بین بلوک‌های منطقی در FPGA ودر نهایت تغییر تابع اجرایی مدار می‌شود.



شکل۱. مدل معماری یک FPGA مبتنی بر SRAM [۱۱]

در همین راستا دو سطح از مقاوم سازی برای ماژول سویچ در مرجع [۱۱] مورد توجه قرار گرفته است. ۱) مقاوم سازی در سطح جعبه سویچ و ۲) مقاوم سازی در سطح ماژول سویچ. که در این دو سطح از کدهای همینگ با قابلیت تصحیح خطای تک بیتی در یک عبارت کدگذاری شده بهره گرفته شده است. در ادامه به توضیح روش‌های ارائه شده در مرجع [۱۱] به صورت دقیق تر پرداخته خواهد شد.



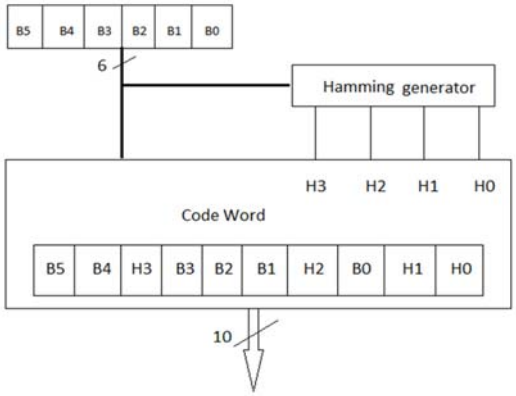
(ب) جعبه سویچ (الف) ماژول سویچ

شکل۲. ساختار یک ماژول سویچ ویک جعبه ی سویچ [۴]

جزییات و کاستیهای روش قبلی

مقاوم سازی در سطح جعبه سویچ

در کد همینگ برای محافظت از N بیت باید تعداد K بیت، به بیت‌های اطلاعات اضافه شود. برای داشتن قابلیت تصحیح یک بیت، N و K باید شرط موجود در رابطه‌ی (۱) را برآورده کنند. بر اساس رابطه (۱) باید ۴ بیت دیگر اضافه شود. شکل۳ ساختار کد مورد نظر را نشان می‌دهد.



شکل۳. ساختار کد همینگ استفاده شده برای هر جعبه‌ی سویچ

$$N+K+1 \leq 2K \qquad (۱)$$

هر جعبه سویچ دارای ۶ بیت برای پیکره‌بندی است، در نتیجه در رابطه (۱) مقدار N=6 است. برای محافظت از ۶ بیت بر اساس رابطه (۱) باید ۴ بیت دیگر اضافه شود. شکل۳ ساختار کد مورد نظر را نشان می‌دهد.

ترکیب حالت‌های ۴ بیت می‌تواند برای کد همینگ استاندارد (۱۱،۱۵) استفاده شود، اما در این کاربرد تنها ۶ بیت در هر جعبه سویچ وجود دارد و کد همینگ استفاده شده (۶،۱۰) است. این نوع کد، کد همینگ کوتاه شده نامیده شده است[۱۲]. با ضرب ماتریس H در کد مورد نظر بردار سندرم به دست می‌آید. اگر بردار سندرم به دست آمده برابر با بیان باینری صفر (۰۰۰۰) باشد، هیچ خطایی در کد رخ نداده است ولی در تمام حالت‌های دیگر، در کد خطا وجود دارد. ماتریس H مربوط به کد همینگ (۶،۱۰) در رابطه (۲) نشان داده شده است (ماتریس شامل اعداد صفر و یک در سمت چپ تساوی، ماتریس H است).

$$\begin{bmatrix}1110000000\\0001111000\\1001100110\\0101010101\end{bmatrix}\begin{bmatrix}B5\\B4\\H3\\B3\\B2\\B1\\H2\\B0\\H1\\H0\end{bmatrix}=\begin{bmatrix}r3\\r2\\r1\\r0\end{bmatrix} \qquad (۲)$$

در عبارت کد شده با کد همینگ اگر خطای تک بیت در کد مورد نظر رخ دهد، سندرم به دست آمده برابر است با ستون باینری متناظر با مکان خطا در ماتریس H. اگر یک خطای دوتایی در کد رخ دهد، سندرم به دست آمده برابر است با XOR بین دو ستون از ماتریس H، که خطا در آنها رخ داده است. به عنوان مثال اگر یک خطای دوتایی در بیت‌های H0 و H1 رخ دهد سندرم به دست آمده برابر است با "۰۰۱۱" که متناظر است با بیان باینری B0 ، در نتیجه کد به اشتباه B0 را تصحیح می‌کند. در خطاهای دوتایی مجاور، تنها اگر خطا در بیت‌های B3 و H3 رخ دهد سندرم خروجی برابر است با "۱۱۱۱"، که با سندرم هیچکدام از خطاهای تک بیتی تداخل ندارد (با هیچکدام از ستون‌های ماتریس H برابر نیست) . در نتیجه این کد در ۸۸/۸۸٪ موارد توانایی تشخیص بین خطا در یک بیت و یا خطا در دو بیت را ندارد و در اکثر موارد، کد به جای تشخیص خطای دو بیتی از تک بیتی، به اشتباه یک بیت دیگر را تصحیح می‌کند.

مقاوم سازی در سطح ماژول سویچ

در این سطح از مقاوم سازی کد همینگ برای محافظت از هر ماژول سویچ اعمال می‌شود. در هر ماژول سویچ ۲۴ بیت کنترلی وجود دارد (N=24) در نتیجه طبق رابطه (۱) تعداد ۵ بیت به عنوان چک بیت باید اضافه شود تا کد قابلیت تصحیح خطاهای تک بیتی را داشته باشد. تعداد ترکیب‌های مختلف حاصل از ۵ بیت می‌تواند برای کد همینگ استاندارد (۲۶،۳۱) استفاده شود، اما در این کاربرد تعداد بیت‌های اطلاعات ۲۴ عدد است. در نتیجه کد استفاده شده، کد همینگ (۲۴, ۲۹) است که این کد نیز یک کد همینگ کوتاه شده است. ماتریس H مربوط به این کد نیز در رابطه (۳) نشان داده شده است. در این ماتریس از خطاهای مجاور دوتایی تنها خطا در مکان ۱۵ و ۱۶ قابل تشخیص است و سندرم متناظر آن (۱۱۱۱۱) با سندرم هیچکدام از خطاهای تک بیتی اشتباه نمی‌شود. در نتیجه این کد در ۹۶/۴۳٪ موارد توانایی تشخیص بین خطا در یک بیت و یا خطا در دو بیت را ندارد و در اکثر موارد، کد به جای تشخیص خطای دو بیتی از تک بیتی، به اشتباه یک بیت دیگر را تصحیح می‌کند.

$$H=\begin{bmatrix}11111111111111000000000000000\\11111100000000111111110000000\\11000^{4}01111000011110000111100\\00110011001100110011001100110\\10101010101010101010101010101\end{bmatrix} \qquad (۳)$$

[4] Selective Bit Placement

راه حل عمومی برای جبران کاستی‌های کار قبلی: استراتژی جایابی بیت انتخابی[۴]

استراتژی جایابی بیت در مرجع [۱۲] برای بهبود تشخیص خطاهای مجاور در حافظه‌ها (برای ۸ ، ۱۶ و ۳۲ بیت اطلاعات) معرفی شده است، اما در این مقاله از این استراتژی برای بهبود تشخیص خطاهای مجاور دوتائی در کد همینگ کوتاه شده (۶،۱۰) و کد همینگ کوتاه شده (۲۴،۲۹) استفاده شده است. از این کدها در مرجع [۱۱] برای مقابله با SEU در ماژول سویچ FPGA های مبتنی بر SRAM استفاده شده است.

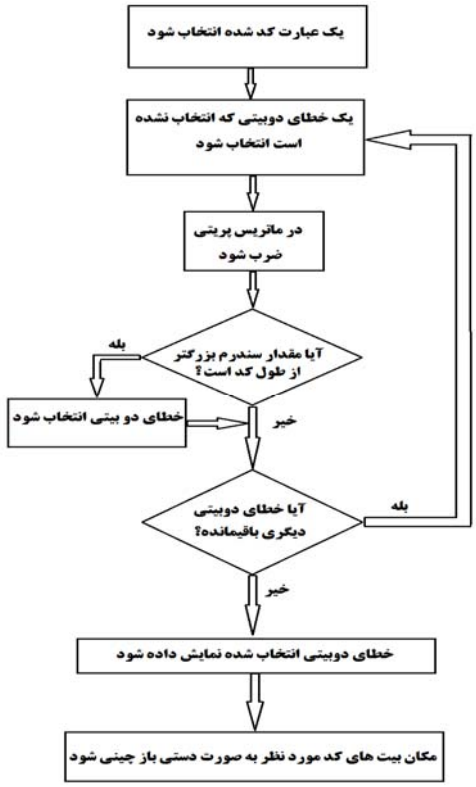
در مقاوم سازی در سطح جعبه سویچ با کد همینگ، ۴ چک بیت به بیت‌های اطلاعات اضافه می‌شود و کد همینگ مورد استفاده، همینگ (۶،۱۰) است. حالت‌های مختلف ۴ بیت می‌تواند ۱۶ حالت متمایز را نشان دهد که یکی از این حالت‌ها برای حالت بدون خطا و ۱۰ حالت برای خطاهای تک بیتی مورد استفاده قرار می‌گیرد، اما ۵ حالت دیگر باقی می‌ماند که بیان باینری آن‌ها بزرگتر از طول کد است و از این حالت‌ها استفاده نشده است. همچنین در مقاوم سازی در سطح ماژول سوئیچ نیز ۵ چک بیت به بیت‌های اطلاعات اضافه می‌شود و کد همینگ مورد استفاده همینگ (۲۴،۲۹) است. حالت‌های مختلف ۵ بیت می‌تواند ۳۲ حالت متمایز را نشان دهد که یکی از این حالت‌ها برای حالت بدون خطا و ۲۹ حالت برای خطا های تک بیتی مورد استفاده قرار می‌گیرد، اما ۲ حالت دیگر باقی می‌ماند که بیان باینری آن‌ها بزرگتر از طول کد است و از این حالت‌ها استفاده نشده است. همچنین خطاهای دوتایی مخصوصی در کد وجود دارند که سندرم آن‌ها با سندرم‌های استفاده نشده برابر است و ترکیب‌های مختلف این خطاها در کد پخش شده است. هدف جایابی بیت انتخابی این است که ابتدا این ترکیب‌های مخصوص را یافته و با استفاده از اطلاعات به دست آمده، بیت‌های کد مورد نظر به گونه‌ای کنار هم قرار داده شوند که قابلیت تشخیص خطاهای مجاور دو تایی در کد افزایش یابد. این مراحل در شکل۴ نشان داده شده است.

روشهای پیشنهادی

جایابی بیت پیشنهادی برای جعبه سویچ

با استفاده از پروسه نشان داده شده در شکل۴ برای همینگ (۱۰،۶)، مکان تمام خطاهای دوتایی مخصوص که سندرم آنها از طول کد بزرگتر است به دست آمده است. این خطاهای دوتایی عبارتند از: ۱۰-۱, ۹-۲, ۸-۳ , ۸-۴, ۹-۴,۱۰-۴, ۸-۵ , ۹-۵, ۱۰-۵, ۸-۶, ۹-۶,۱۰-۶, ۸-۷,۸-۷,۹-۷,۱۰-۷.

تنها ترکیبی از خطاهای دوتایی مجاور که در حالت عادی، سندرم آن با سندرم خطاهای تک بیت تداخل ندارد" ۷-۸" (H3-B3) است، بنا بر این تنها یک خطای دوتایی مجاور از ۹ خطای دوتایی مجاور با استفاده از جایابی بیت مورد استفاده در مقاله [۱۱] قابل تشخیص است و سندرم آن از سندرم خطاهای تک بیت و از حالت بدون خطا، متمایز است. با توجه به اطلاعات فوق جایابی بیت پیشنهادی در جدول۱ آمده است و ماتریس H مربوط به آن در رابطه (۴) نشان داده شده است (ماتریس شامل اعداد صفر و یک در سمت چپ تساوی، ماتریس H است).



شکل۴. پروسه جایابی بیت انتخابی

$$\begin{bmatrix}0001001010\\1110000100\\0010110110\\0100101101\end{bmatrix}\begin{bmatrix}H2\\B1\\B2\\H3\\B0\\H1\\B4\\B3\\B5\\H0\end{bmatrix}=\begin{bmatrix}r3\\r2\\r1\\r0\end{bmatrix} \quad (۴)$$

در جایابی بیت پیشنهادی به غیر از ترکیب‌های ۲-۳ ، ۶-۵ و ۴-۵ تمام ترکیب‌های دوتایی مجاور با ترکیب‌های مخصوص که

سندرم آنها از طول کد بزرگتر است مطابق هستند. در نتیجه ۶ خطای مجاور از ۹ خطا قابل تشخیص است.

جدول۱. جایابی بیت پیشنهادی برای همینگ (۱۰،۶)

H2	B1	B2	H3	B0	H1	B4	B3	B5	H0
۴	۵	۶	۸	۳	۲	۹	۷	۱۰	۱

جایابی بیت پیشنهادی برای ماژول سویچ

با استفاده از پروسه نشان داده شده در شکل۴ برای همینگ (۲۹،۲۴)، مکان تمام خطاهای دوتایی مخصوص که سندرم آنها از طول کد بزرگتر است به دست آمده است. این خطاهای دوتایی عبارتند از:

۱۶-۱۴، ۱۵-۱۶، ۱۴-۱۷، ۱۵-۱۷، ۱۲-۱۸، ۱۳-۱۸، ۱۲-۱۹، ۱۳-۱۹، ۱۰-۲۰، ۱۱-۲۰، ۱۰-۲۱، ۱۱-۲۱، ۸-۲۲، ۹-۲۲، ۸-۲۳، ۹-۲۳، ۶-۲۴، ۷-۲۴، ۶-۲۵، ۷-۲۵، ۴-۲۶، ۵-۲۶، ۴-۲۷، ۵-۲۷، ۲-۲۸، ۳-۲۸، ۲-۲۹، ۳-۲۹ .

تنها ترکیبی از خطا های دوتایی مجاور که در حالت عادی، سندرم آن با سندرم خطاهای تک بیت تداخل ندارد" ۱۵-۱۶" است، بنا بر این تنها ۱ خطای دوتایی مجاور از ۲۸ خطای دوتایی مجاور با استفاده از جایابی بیت مورد استفاده در مقاله [۱۱] قابل تشخیص است و سندرم آن از سندرم خطاهای تک بیت و از حالت بدون خطا، متمایز است.

با توجه به اطلاعات فوق جایابی بیت پیشنهادی در شکل ۵ آمده است و ماتریس H مربوط به آن نیز در رابطه (۵) نشان داده شده است.

$$H=\begin{bmatrix}1010101010101010101010101010100\\0101010101010101011010101010100\\0101010110101010010101010110100\\0101101001011010010110100101010\\0110011001100110011001100110101\end{bmatrix} \quad (۵)$$

در جایابی بیت پیشنهادی به غیر از ترکیب‌های ۱-۲ ، ۲۸-۴، ۲۶-۶، ۲۴-۸، ۲۲-۱۰، ۲۰-۱۲ و ۱۸-۱۴ تمام ترکیب‌های دوتایی مجاور با ترکیب‌های مخصوص که سندرم آنها از طول کد بزرگتر است مطابق هستند. در نتیجه ۲۱ خطای مجاور از ۲۸ خطا قابل تشخیص است.

[۱۶ - ۱۵ - ۱۷ - ۱۴ - ۱۸ - ۱۳ - ۱۹ - ۱۲ - ۲۰ - ۱۱ - ۲۱-۱۰ - ۲۲ - ۹ - ۲۳ - ۸ - ۲۴ - ۷ - ۲۵ - ۶ - ۲۶ - ۵ - ۲۷ - ۴ - ۲۸ - ۳- ۲۹ - ۲ - ۱]

شکل۵. جایابی بیت پیشنهادی برای همینگ (۲۴،۲۹)

روش پیشنهادی مبتنی بر الگوریتم ژنتیک

همانطور که در قسمت‌های قبلی بیان شد تعدادی سندرم آزاد در کدهای همینگ کوتاه شده وجود دارد که آنها با هیچکدام از خطاهای تک بیتی و سندرم حالت بدون خطا تداخل ندارند. استراتژی جایابی بیت از آن‌ها برای افزایش قابلیت تشخیص خطاهای مجاور استفاده می‌کند. اما از نقاط ضعف این روش می‌توان به بازچینی دستی در پایان الگوریتم اشاره کرد، که در تعدادی از کدها، یافتن جواب مناسب و بهینه به صورت دستی منطقی نیست و ممکن است به دلیل خستگی و سخت بودن کار و موارد دیگر، نتیجه نهایی به جوابی که می‌توان به آن دست یافت منتهی نشود. همچنین از دیگر نقاط ضعف تکنیک جایابی بیت می‌توان به ماتریس توازن آن اشاره کرد که ستون-های آن فقط شامل بیان باینری اعدادی کوچکتر و برابر با طول کد است و سندرم‌های آزاد آن بزرگتر از طول کد هستند، در صورتی که منعی برای استفاده از سندرم‌های بزرگتر از طول کد در ماتریس توازن و آزاد کردن سندرم‌های دیگر وجود ندارد.

در این بخش جهت رفع مشکلات جایابی بیت، روشی مبتنی بر الگوریتم ژنتیک ارایه شده است. الگوریتم ژنتیک با توجه به سندرم‌های آزاد در کد همینگ کوتاه شده به دنبال تولید ماتریسی است که در آن قابلیت تشخیص خطاهای مجاور دوتائی افزایش یافته باشد. برای این منظور تابع برازندگی مربوط باید شرایط موجود در جدول ۲ را برآورده کند.

جدول۲. شروط اعمال شده برای تولید ماتریس H

شماره شرط بر اساس تقدم	بیان شرط	مقدار جریمه
۱	اعدادی که توانی از ۲ هستند باید در ماتریس وجود داشته باشند. (فرض کنید هر ستون ماتریس از باینری به دهدهی تغییر داده شده است.)	۲/۷۵
۲	اعدادی که توانی از ۲ هستند نباید در ماتریس تکراری باشند. (فرض کنید هر ستون ماتریس از باینری به دهدهی تغییر داده شده است.)	۲/۵
۳	یک ستون که تمام مقادیر آن صفر است نباید در ماتریس وجود داشته باشد.	۲/۲۵
۴	XOR بین دو ستون مجاور در ماتریس نباید یک ستون که تمام مقادیر آن صفر است بشود.	۲
۵	هر ستون در ماتریس باید منفرد باشد	۱/۷۵
۶	XOR بین دو ستون مجاور در ماتریس نباید با ستون های ماتریس برابر باشد .	۱/۲۵

دلیل شرط‌های در نظر گرفته شده در جدول۲ عبارتند از: شرط ۱ حضور ستون‌های مربوط به چک بیت ها را در ماتریس H تضمین می‌کند. شرط ۲ عدم تکرار این ستون‌ها را در ماتریس H تضمین می‌کند. شرط ۳ عدم تداخل خطاهای تک بیتی را با حالت بدون خطا تضمین می‌کند. شرط ۴ عدم تداخل خطاهای دو بیتی مجاور را با حالت بدون خطا تضمین می‌کند. شرط ۵ قابلیت تصحیح خطاهای تک بیتی را تضمین می‌کند و در نهایت شرط ۶ قابلیت تشخیص خطاهای دو بیتی مجاور را تضمین می‌کند.

بیت‌های مربوط به ماتریس H به عنوان کروموزوم برای الگوریتم ژنتیک در نظر گرفته شده است. به عنوان مثال در کد همینگ (۱۰,۶) ۴۰ بیت (۴۰=۴×۱۰) در ماتریس H وجود دارد، در نتیجه کروموزوم مربوط به آن یک رشته ۴۰ بیتی است. تابع برازندگی نیز مبتنی بر شرط‌های موجود در جدول ۲ تعریف می شود. به ازای عدم برقراری هر شرط در جدول ۲، جریمه در نظر گرفته شده برای آن شرط به تعداد رعایت نشدن آن شرط به تابع برازندگی اعمال می‌شود. مقدار جریمه (مقدار این جریمه‌ها به صورت تجربی قرار داده شده است) مربوط به هر شرط در جدول ۲ آمده است. همچنین برای استفاده از الگوریتم ژنتیک از نرم افزار MATLAB استفاده شده است. پارامترهای الگوریتم ژنتیک نیز در جدول ۳ ارائه شده است.

نتایج شبیه سازی

روش ارائه شده در قسمت قبل بر روی کدهای همینگ (۶ ,۱۰) و همینگ (۲۴, ۲۹) اعمال شده است. ماتریس H مربوط به کد همینگ (۶ ,۱۰) در رابطه (۶) نشان داده شده است (ماتریس شامل اعداد صفر و یک در سمت چپ تساوی، ماتریس H است).

$$\begin{bmatrix}0010011010\\1011001110\\0101001001\\0011100101\end{bmatrix}\begin{bmatrix}H2\\H1\\D5\\D4\\H0\\H3\\D3\\D2\\D1\\D0\end{bmatrix}=\begin{bmatrix}r3\\r2\\r1\\r0\end{bmatrix} \qquad (۶)$$

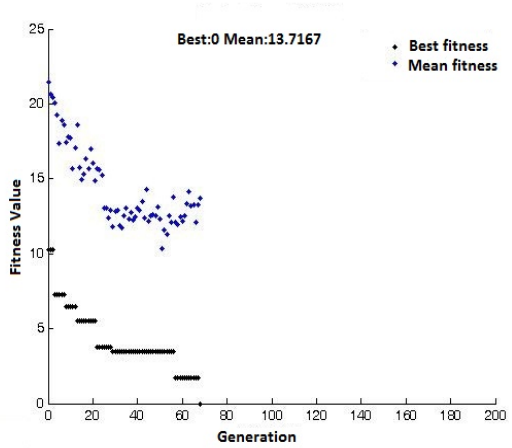
فرض کنید هر ستون ماتریس H رابطه (۶) از یک ستون باینری به یک عدد دهدهی تبدیل شود . در نتیجه دنباله اعداد مقابل، ماتریس H رابطه (۶) را توصیف می‌کند: [۳ ، ۱۲ ، ۵ ، ۱۴ ، ۸ ، ۱ ، ۷ ، ۱۳ ، ۲ ، ۴]. حال ماتریس دیگری را در نظر بگیرید که از XOR بین ستون‌های مجاور دوتائی ماتریس H به دست آمده است. هر ستون این ماتریس را نیز به اعداد دهدهی تبدیل کنید. دنباله اعداد روبرو این ماتریس را توصیف می‌کنند: [۱۵ ، ۹ ، ۱۱ ، ۶ ، ۹ ، ۶ ، ۱۰ ، ۱۵ ، ۶]. همانطور که می‌توان دید هیچکدام از اعداد در دنباله دوم (اعداد حاصل از XOR بین ستون‌های دوتائی مجاور) با هیچکدام از اعداد دنباله اول (ستون‌های ماتریس H) برابر نیستند و تمام خطاهای دوتائی مجاور هم زمان قابل تشخیص است. همچنین در شکل ۶ نیز مقدار تابع برازندگی در هر نسل برای کد همینگ (۶ , ۱۰) نشان داده شده است که در نهایت به برازندگی صفر رسیده است و این امر تشخیص تمام خطاهای دو بیتی مجاور را در این کد نشان می دهد.

الگوریتم ژنتیک برای کد همینگ (۲۴, ۲۹) به دلیل کم بودن تعداد سندرم‌های آزاد (۲ سندرم آزاد) نسبت به طول کد نتوانست ماتریس H معتبری ارائه دهد و در ماتریس ارائه شده شرط ۲ از جدول ۲ برقرار نبود.

در جدول ۴ نتایج حاصل از جایابی بیت پیشنهادی برای کد همینگ (۲۴ , ۲۹) با مرجع [۱۱] مقایسه شده است. در جدول ۵ نیز نتایج حاصل از جایابی بیت پیشنهادی و الگوریتم ژنتیک برای کد همینگ (۶ , ۱۰) با مرجع [۱۱] مقایسه شده است. در جدول ۴ و ۵ اعدادی که زیر آن‌ها خط کشیده شده است نشان دهنده تداخل (برابر بودن سندرم‌های مربوطه) بین خطاهای مجاور دوتائی و خطاهای تک بیتی در ستون های مربوطه است.

در جدول۴ تعداد گیت‌های XOR دو ورودی که برای کد گذاری و کد برداری مورد نیاز است از ماتریس H و با استفاده از رابطه (۷) محاسبه شده است [۱۳].

$$\sum_{\#rows}(row\ weight-1)=\#2-input\ XOR\ gate \qquad (۷)$$



شکل۶ . مقدار تابع برازندگی در هر نسل برای کد همینگ (۶ , ۱۰)

جدول ۳. مقدار پارامترهای استفاده شده در الگوریتم ژنتیک.

نوع پارامتر	تابع انتخاب	تابع عملگر جهش	نرخ جهش	تابع عملگر Crossover	تعداد جمعیت	بهترین مقدار تابع برازندگی
مقدار پارامتر	یکنواخت اتفاقی	یکنواخت	۰/۲۴	پراکنده	۴۵	۰

جدول۴. مقایسه جایابی بیت پیشنهادی برای کد همینگ (۲۴ , ۲۹) با مرجع [۱۱]

	ستون های ماتریس H	ماکزیمم عمق logic	تعداد گیتهای XOR دو ورودی	XOR بین ستونهای مجاور دوتائی در ماتریس H	تعداد خطاهای مجاور دوتائی هم زمان قابل تشخیص نسبت به تمام خطاهای دوتائی مجاور	درصد خطاهای مجاور دوتائی هم زمان قابل تشخیص
مرجع [۱۱]	[۱ – ۲ – ۳- ۴ – ۵ – ۶ – ۷ – ۸ – ۹ – ۱۰ – ۱۱ – ۱۲ – ۱۳ – ۱۴ – ۱۵ – ۱۶ – ۱۷ – ۱۸ – ۱۹ – ۲۰ – ۲۱ – ۲۲ – ۲۳ – ۲۴ – ۲۵ – ۲۶ – ۲۷ – ۲۸ – ۲۹]	۴	۶۶	[۳ – ۱ – ۷ – ۱ – ۳ – ۱ – ۱۵ – ۱ – ۳ – ۱ – ۷ – ۱ – ۳ – ۱ – ۳۱ – ۱ – ۳ – ۱ – ۷ – ۱ – ۱۵ – ۱ – ۳ – ۱ – ۷ – ۱]	۱/۲۸	۳٪/۵۷
جایابی بیت پیشنهادی	[۱ – ۲ – ۲۹- ۳- ۲۸– ۴ –۲۷– ۵ – ۲۶ – ۶ – ۲۵ – ۷ – ۲۴ – ۸ – ۲۳ – ۹ – ۲۲ – ۱۰ – ۲۱ – ۱۱ – ۲۰ – ۱۲ – ۱۹ – ۱۳ – ۱۸ –۱۴ – ۱۷ – ۱۵ – ۱۶]	۴	۶۶	[۳ – ۳۱ – ۳۰ – ۳۱ – ۲۴ – ۳۱ – ۳۱ – ۳۱ – ۳۱ – ۲۴ – ۳۱ – ۳۰ – ۳۱ – ۱۶ – ۳۱ – ۳۰ – ۳۱ – ۳۰ – ۳۱ – ۲۸ – ۳۱ – ۳۰ – ۳۱ – ۲۴ – ۳۱ – ۳۰ – ۳۱ – ۲۸ – ۳۱ – ۳۰ – ۳۱ – ۳۱]	۲۱/۲۸	۷۵٪

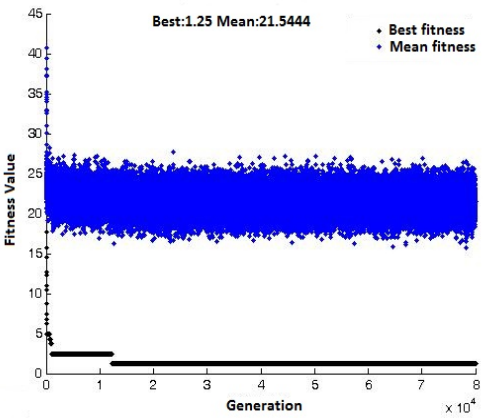
جدول ۵. مقایسه جایابی بیت پیشنهادی و الگوریتم ژنتیک برای کد همینگ (۶ , ۱۰) با مرجع [۱۱]

	ستون های ماتریس H	ماکزیمم عمق logic	تعداد گیتهای XOR دو ورودی	XOR بین ستونهای مجاور دوتائی در ماتریس H	تعداد خطاهای مجاور دوتائی هم زمان قابل تشخیص نسبت به تمام خطاهای دوتائی مجاور	درصد خطاهای مجاور دوتائی هم زمان قابل تشخیص
مرجع [۱۱]	[۱- ۲ – ۳ – ۴ – ۵ - ۶ – ۷ – ۸ – ۹ – ۱۰]	۳	۱۳	[۳ – ۱ – ۷ – ۱ – ۳ – ۱ – ۱۵ – ۱ – ۳]	۱/۹	۱۱٪/۱۱
جایابی بیت پیشنهادی	[۱ – ۱۰ – ۷ – ۹ – ۲ – ۳ – ۸ – ۶ – ۵ – ۴]	۳	۱۳	[۱۰ – ۱۵ – ۱۴ – ۱۱ – ۱ – ۱۲ – ۱۴ – ۳ – ۱]	۶/۹	۶۶٪/۶۶
ماتریس H تولید شده با الگوریتم ژنتیک (بدون در نظر گرفن شرط افزونگی برای تعداد گیت های XOR)	[۳ – ۱۲ – ۵ – ۱۴ – ۸ – ۱ – ۷ – ۱۳ – ۲ – ۴]	۳	۱۵	[۱۵ – ۹ – ۱۱ – ۶ – ۹ – ۶ – ۱۰ – ۱۵ – ۶]	۹/۹	۱۰۰٪
ماتریس H تولید شده با الگوریتم ژنتیک (با در نظر گرفتن شرط افزونگی برای تعداد گیت های XOR)	[۱۰ – ۲ – ۵ – ۸ – ۴ – ۳ – ۱۴ – ۹ – ۶ – ۱]	۳	۱۳	[۸ – ۷ – ۱۳ – ۱۲ -۷- ۱۳- ۷- ۱۵- ۷]	۸/۹	۸۸٪/۸۸

در رابطه (۷) منظور از وزن هر سطر (row weight) تعداد یک-های موجود در آن سطر است. با استفاده از تعداد گیت‌های XOR سربارهای مربوط به حجم سخت افزاری قابل محاسبه است. مقدار تاخیر در کدگذاری و کد برداری نیز با ماکزیمم عمق logic در مدار کد گزار و کد بردار مرتبط است که با استفاده از رابطه (۸) از روی ماتریس H به دست می آید[۱۳].

$$log_2(max.1's\ in\ any\ row) = maximum\ logic\ depth \quad (۸)$$

همانطور که در جدول ۵ نشان داده شده است تعداد گیت‌های XOR زمانی که از الگوریتم ژنتیک استفاده شده است (بدون شرط در نظر گرفته شده برای تعداد گیت‌های XOR) افزایش یافته است که نشان دهنده افزونگی سخت افزاری در مدار است. برای جلوگیری از این افزونگی، یک شرط دیگر باید به شرط‌های جدول ۲ افزود، که به ازای هر گیت XOR اضافه شده، جریمه ای به مقدار " ۱" در تابع برازندگی اعمال شود. حال با توجه به شرط اضافه شده پس از اعمال الگوریتم ژنتیک همانطور که در جدول ۵ نشان داده شده است افزونگی ایجاد شده در گیت‌های XOR از بین می‌رود اما درصد تشخیص خطاهای مجاور دوتائی هم زمان نسبت به قبل کاهش می‌یابد که نشان دهنده مصالحه بین کارآیی و افزونگی در روش پیشنهادی مبتنی بر الگوریتم ژنتیک است. در شکل ۷ مقدار تابع برازندگی در هر نسل برای کد همینگ (۶ , ۱۰) با در نظر گرفتن شرط مربوط به تعداد گیت‌های XOR نشان داده شده، که در نهایت به برازندگی ۱/۲۵ رسیده است که با توجه به جدول۲ نشان دهنده عدم تشخیص یک خطای مجاور دو بیتی از یک خطای تک بیتی است.



شکل۷ . مقدار تابع برازندگی در هر نسل برای کد همینگ (۶ , ۱۰) با در نظر گرفتن شرط عدم افزونگی در تعداد گیتهای XOR

نتیجه گیری

استفاده از FPGA های مبتنی بر SRAM در کاربردهای فضایی مانند عملیات اکتشاف دور به دلیل قابلیت باز پیکره‌بندی که در این قطعات قرار داده شده است بسیار ضروری است. نتایج آزمون‌های به دست آمده بر روی FPGA های مبتنی بر SRAM نشان می‌دهد که این قطعات فوق العاده به تشعشعات فضایی حساس هستند و نرخ SEU در آن‌ها بسیار زیاد است. کد همینگ با قابلیت تصحیح خطای تک بیتی، برای مقابله با SEU در ماژول سویچ FPGA های مبتنی بر SRAM استفاده شده است.

در همین راستا این مقاله دو جایابی بیت جدید برای بیت-های پیکره بندی ماژول سویچ ارائه می‌دهد که موجب افزایش قابلیت تشخیص خطاهای مجاور دوتائی هم زمان (در مقاوم سازی با کد همینگ در سطح جعبه سویچ و در سطح ماژول سویچ) می‌شوند. همچنین یک روش دیگر مبتنی بر الگوریتم ژنتیک نیز ارائه شده است که اعمال آن (در مقاوم سازی با کد همینگ در سطح جعبه سویچ) موجب افزایش تشخیص خطاهای مجاور دوتائی می‌شود. هر دو روش اعمال شده (جایابی بیت انتخابی و روش ارائه شده مبتنی بر الگوریتم ژنتیک) هیچگونه افزونگی را تحمیل نمی‌کنند. البته در روش ارائه شده مبتنی بر الگوریتم ژنتیک می‌توان با افزایش تعداد گیت‌های XOR مورد نیاز برای کدگذاری / کد برداری تمام خطاهای مجاور دوتائی هم زمان را تشخیص داد که نشان دهنده مصالحه بین افزونگی سخت افزاری و کارآیی در این روش است.

برای کارهای آتی می توان به کاربرد جایابی بیت انتخابی و الگوریتم ژنتیک در کدهای دیگر پرداخت.

مراجع

[1] A.Raoul Velazco and B. Pascal Fouillat, “Radiation Effects on Embedded Systems”, Springer Netherlands, Page(s): 1-9, 2007.

[2] Gregory Allen, Gary Swift and Carl Carmichael, “Vitex-4 VQ Static SEU Characterization Summary”, JPL Publication 08-16 4/08, 2008.

[3] Niknahad, Mahtab, Oliver Sander, and Juergen Becker. "Fine grain fault tolerance—A key to high reliability for FPGAs in space." Aerospace Conference, 2012 IEEE. IEEE, 2012.

[4] Alireza Rohani, Hamid R.zarandi and Mahroo Zandrahimi, “New Switch Box Architecture for SEU Detection in SRAM-based FPGAs”,

2nd International Conference on Communication, Networking & Broadcasting ; Computing & Processing (Hardware/Software) Computer Science and its Applications,2009 IEEE.

[5] E. Ibe, H. Taniguchi, Y. Yahagi, K. Shimbo, and T. Toba, "Impact of scaling on neutron-induced soft error in SRAMs from a 250 nm to a 22 nm design rule," IEEE Trans. Electron Devices, vol. 57, no. 7, pp. 1527–1538, Jul. 2010.

[6] R. Baumann, "Soft errors in advanced computer systems", IEEE Trans. Device Mater. Reliab., vol. 22, no. 3, pp. 258–266, May–Jun. 2005.

[7] R. K. Lawrence and A. T. Kelly, "Single event effect induced multiple-cell upsets in a commercial 90 nm CMOS digital technology," IEEE Trans.Nucl. Sci., vol. 55, no. 6, pp. 3367–3374, Dec. 2008.

[8] S. Satoh, Y. Tosaka, and S. A. Wender, "Geometric effect of multiple-bit soft errors induced by cosmic ray neutrons on DRAM's," IEEE Electron Device Lett., vol. 21, no. 6, pp. 310–312, Jun. 2000.

[9] F. Lima, G. Neuberger, R. Hentschke, "Designing Fault-tolerant Techniques for SRAM-Based FPGAs", IEEE Design & Test of Computers, vol. 21, pp. 552-562, 2004.

[10] M. Zhu, L.Xiao,S.Li, and Y.Zhang, "Efficient two-dimensional error codes for multiple bit upsets mitigation in memory," in Proc. IEEE 25th Int. Symp. Defect Fault Tolerance VLSI Syst., 2010, pp. 129–135.

[11] A. Rohani and H. R. Zarandi, "Mitigating and Tolerating SEU Effects in Switch Modules of SRAM-based FPGAs", 5th Southern Conference on Computing & Processing (Hardware/Software) Programmable Logic, 2009 IEEE.

[12] Alfonso Sánchez-Macián, Pedro Reviriego, and Juan Antonio Maestro, "Enhanced Detection of Double and Triple Adjacent Errors in Hamming Codes Through Selective Bit Placement", IEEE Transactions On Device and Materials Reliability, Vol. 12, No. 2, JUNE 2012.

[13] A. Dutta and N. A. Touba, "Multiple bit upset tolerant memory using a selective cycle avoidance based SEC-DED-DAEC code," in Proc. 25th IEEE VLSI Test Symp., May 2007, pp. 349–354.