تکنیک جدید مبتنی بر SystemC جهت تزریق خطا در مداری دیجیتال ترکیبی

رضا امیدی قوشه‌بلاغ[۱]، کریم محمدی[۲]
۱- دانشگاه علم و صنعت ایران، rezaomidi@iust.ac.ir
۲- دانشگاه علم و صنعت ایران، mohammadi@iust.ac.ir

چکیده

در طراحی‌های نظامی و هوانوردی غالبا طراحان برای افزایش قابلیت اطمینان سیستم از روش‌های افزونگی یدکی یا موقت مانند افزونگی دوگانه یا افزونگی سه‌تایی استفاده می‌کنند مشکل این تکنیک‌ها هزینه اضافی می‌باشد که به سیستم تحمیل می‌گردد و طراح باید بین هزینه تحمیلی و قابلیت اطمینان حصول شده بتواند قیاس انجام دهد. بنابراین آنالیز عددی قابلیت اطمینان از جنبه‌های ضروری طراحی در این کاربردها می‌باشد. در این مقاله تکنیکی مبتنی بر کتابخانه SystrmC جهت تزریق خطا ارائه شده است این تکنیک با توجه به قابلیت ادغام بستر تست در مدار تحت تست به مراتب از سایر روش‌های موجود از نظر توسعه راحتر و از لحاظ زمان شبیه‌سازی سریعتر می‌باشد. این تکنیک در مدارهای محک اجرا و نتایج در این مقاله گزارش گردیده است.

واژه‌های کلیدی: *قابلیت اطمینان - تزریق خطا - SystemC*

مقدمه

تزریق خطا برای ارزیابی درصد پوشش خطا و قابلیت اطمینان سیستم‌های هدف استفاده می‌شود. از بین روش‌های مختلف تزریق خطا روش شبیه‌سازی به دلیل انعطاف‌پذیری، مشاهده‌پذیری و کنترل‌پذیری بالا به صورت گسترده مورد استفاده قرار می‌گیرد. شبیه‌سازی در سطوح مختلف گیت، رجیستر و سیستم قابل اجراست. شبیه‌سازی تزریق خطا در سطح گیت قادر به آنالیز مشخصه و خطای المان‌های آی‌سی می‌باشد. اما این روش دارای محدودیت‌هایی مانند طولانی بودن زمان شبیه‌سازی و سختی توسعه برای مدارهای بزرگ می‌باشد. یکی از دلایل این امر عدم ترکیب بستر تست در محیط زبان توصیف سخت‌افزار می‌باشد. بستر تست که یک زبان سطح بالا است جهت تزریق خطا مستلزم شبیه‌سازی مستقیم زبان سخت‌افزار یا فراخوانی برنامه شبیه‌ساز زبان مذکور می‌باشد. شبیه‌سازی مستقیم زبان سخت‌افزار در بستر تست توسعه مدار را مشکل‌تر می‌کند و نمی‌توان از مزیت سلسه مراتبی و ماژولار بودن برنامه‌های سخت‌افزار استفاده کرد. روش فراخوانی زبان شبیه‌ساز سخت افزار از طرف بستر تست مانند فراخوانی ModelSim از محیط MATLAB مستلزم جلسه کاری است که این امر زمان شبیه‌سازی تزریق خطا را بسیار افزایش می‌دهد[۶]. البته ابزارهایی نیز جهت تزریق خطا در محیط VHDL مانند GSTF مورد استفاده قرار می‌گیرد که نتایج گزارش شده برای تکنیک mutant نیز نشان دهنده زمان طولانی شبیه‌سازی است [۷]. کتابخانه SystemC اساسا برای رفع نقص برنامه C جهت شبیه‌سازی بخش‌های زبان سخت‌افزار برای سیستم در تراشه (SOC) تعبیه شده است در این مقاله با استفاده از کتابخانه SystemC روشی بر مبنای تکنیک اصلاح کد ارائه شده است. این روش قابلیت توسعه به صورت سلسه‌مراتبی را داراست و با توجه به اینکه نیازی به جلسه‌کاری بین دو نرم‌افزار نیست دارای سرعت بیشتری است و توسعه آن در مدارهای بزرگ‌تر به راحتی امکان‌پذیر می‌باشد. در ادامه این مقاله می‌خوانید: مدارهای خودرس و دوپلکس، روش‌ها شبیه‌سازی خطا، محیط تزریق خطا بر مبنای SystemC، نتایج و چیدمان پیاده‌سازی.

مدارهای خود رس (Self Checking) و دوپلکس (Duplex)

آشکارسازی هم‌زمان خطا تایید کننده نتایج حاصله از مدار طی عملکرد عادی آن می‌باشد. این نوع آشکارسازی به واسطه دونسخه‌نویسی و مقایسه قابل اجراست اما این تکنیک مستلزم سربار بیش از ۱۰۰ درصد می‌باشد. برای دست‌یابی به طرحی با سربار سخت‌افزاری کمتر، طراحی مدار به صورت خودرس می‌تواند انتخاب مناسبی باشد. در طراحی خودرس، یک مدار پیچیده به بلوک‌های تابعی تشکیل‌دهنده آن بخش‌بندی می‌شود و هر یک از این بلوک‌ها مطابق ساختار شکل ۳ پیاده‌سازی می‌گردد. این ساختار بلوک‌های تابعی را به گونه‌ای پیاده‌سازی می‌کنند که خروجی‌های حاصله متعلق به یک کد آشکارسازی خطا باشد تا بخش چک‌کننده که بر مبنای آن آشکارسازی هم‌زمان خطا را انجام می‌دهند.

هدف غایی از طراحی مدارهای خودرس رسیدن به حالتی است که تحت عنوان TSC(Totally Self Checking) بیان می‌شود. رسیدن به این هدف مستلزم این است که تحت هر خطای مدل شده، اولین خروجی معیوب و خطادار بلوک تابعی، در خروجی چک‌کننده خود را نشان دهد. برای دستیابی به این هدف بلوک تابعی باید دارای برخی ویژگی‌ها باشد.

ایمن در برابر خطا(Fault Secure): در برابر هر خطای مدل شده، خروجی‌هایی خطادار متعلق کد خروجی نباشد تا بر اساس آن بلوک چک کننده خطا را تشخیص دهد.

خودآزمایی(Self Testing): برای هر خطای مدل شده یک بردار ورودی طی عملکرد نرمال مدار وجود دارد که بردار خروجی آن متعلق به کد خروجی نیست.

این ویژگی از خطاهای بیهوده اجتناب می‌کند چنین خطاهایی در مدار به صورت غیر قابل آشکارساز باقی می‌مانند و می‌توانند در ترکیب با خطاهای جدید در مدار، منجر به خطاهای چندگانه شوند که می‌تواند خاصیت ایمنی در برابر خطا را از بین ببرد. بنابراین ترکیب خواص خودآزمایی و ایمنی در برابر خطا (خاصیت TSC) بالاترین سطح از حفاظت را داراست.

TSC: مدار هم ایمن در برابر خطاست هم خود آزما.

خاصیت ایمنی در برابر خطا بسیار مهم‌تر از خودآزمایی است چرا که این خاصیت آشکارسازی هر تک خطایی را ضمانت می‌کند اما دستیابی به آن سخت‌تر است. خاصیت خودآزمایی، به ویژه برای خطاهای انباشتگی به آسانی قابل دستیابی است چرا که برای حذف خطاهای حشو کافی است مدار ساده‌سازی شود.

پس از طراحی مدار با استفاده ار تکنیک‌های تحمل‌پذیری خطا ارزیابی خاصیت آن (ایمن در برابر خطا، خودآزمایی و ...) مستلزم استفاده از روش‌های تزریق یا آنالیز خطا می‌باشد. در ادامه نگاهی اجمالی به روش‌های تزریق خطا- بحث اصلی این مقاله- خواهیم داشت.

روش‌های شبیه‌سازی خطا

تکنیک‌های شبیه‌سازی خطا بر مبنای ساختارهای تغییرپذیر، ساختارهای خرابکار و شبیه‌سازی دستوری قابل اجراست. مباحث الگوسازی محیط و ایجاد محیط واقعی جهت تزریق خطا خارج از بحث این مقاله می‌باشد. در ادامه تکنیک‌های شبیه‌سازی خطا توضیح داده شده‌اند.

تکنیک دستور شبیه‌ساز

تکنیک دستور شبیه‌ساز (Simulator Command) از دستورات درون-ساخت شبیه‌ساز جهت تزریق خطا در متغیرها و سیگنال‌های مدل تحت تست استفاده می‌کند. برای این منظور یک‌سری دستورات کاذب در شبیه‌ساز در نظر گرفته می‌شود. این دستورات در ورژن ۲.۲ کتابخانه SystemC وجود دارد[۳]. این روش نیاز به اصلاح و تغییر در مدل ندارد و از این نظر ساده‌تر می‌باشد و نسبت به روش‌های دیگر تعداد تکرارهای کمتری نیاز دارد. با این وجود برای این روش نیز محدودیت‌هایی -از جمله در ارائه مدل‌های مختلف خطا در متغیر- دائمی گزارش شده است[۴]. علاوه بر آن این تکنیک به شدت به قابلیت‌های شبیه‌ساز در ایجاد و مدیریت دستورات کاذب وابسته است[۵].

تکنیک ساختارهای تغییرپذیر و خرابکار

خرابکار (Saboteur) المان خاصی است که بین درایو و دریافت-کننده سیگنال قرار داده می‌شود و مشخصه این سیگنال از جمله منطق و زمان‌بندی آن را دست‌خوش تغییر می‌کند. در این روش بخش داخلی گیت‌ها و رجیسترها بدون تغییر باقی می‌مانند و صرفا سیگنال‌های بین المان‌ها متاثر خواهند شد بنابراین با افزایش تعداد سیگنال‌ها پیچیدگی این روش بیشتر خواهد شد. این روش مستلزم تشخیص سیگنال‌دهی ماژول هدف می‌باشد[۳ و۶]. همچنین با ساختار سلسه مراتبی و ماژولار مورد استفاده در زبان‌های توصیف سخت‌افزار تطابق ندارد.

از سوی دیگر ساختارهای تغییرپذیر (mutant) شکل خاصی از توصیف المان‌های مداری است که در آن توصیف معمول المان‌ها با توصیفی جایگزین می‌شود به طوری که قابلیت تزریق خطا در خود المان فراهم می‌آید و این خطا در خروجی خود را نشان می‌دهد و المان‌هایی که از آن درایو می‌شوند متاثر خواهند شد. این روش در محیط HDL در مرجع [۵] و در محیط SystemC به صورت حالت خاص در کارت‌های هوشمند اجرا شده است. روش ارائه شده در این مقاله ساختارهای تغییرپذیر در محیط SystemC می‌باشند که مشابه محیط HDL توسعه پیدا می‌کنند بنابراین مزیت سلسله مراتبی HDL و قابلیت‌های بستر تست C را در هم تلفیق می‌کند.

محیط ارائه شده بر مبنای SystemC جهت تزریق خطا

جهت تزریق خطا در محیط SystemC ساختار شکل ۲ ترتیب داده شده است. این ساختار شامل کتابخانه المان‌ها، بستر تست، مدار تحت تست، مدار موید و مقایسه‌کننده می‌باشد. در ادامه هر یک از بخش‌ها به طور مختصر توضیح داده شده است.

کتابخانه المان‌ها: ابتدا هر یک از المان‌های مورد نیاز در مدار تحت تست تعریف و در کتابخانه قرار داده می‌شود این المان‌ها شامل توابع پایه مانند AND، OR، XOR و ... می‌باشند و پس از تعریف و افزوده شدن به کتابخانه به هر تعداد قابل جایگزینی (Instantiation) در هر یک از بخش‌های مدار تحت تست، مدار موید و مقایسه کننده می‌باشد. این شکل از جایگزینی مشابه توصیف ساختاری زبان‌های HDL می-

باشد و می‌توان با استفاده از ابزارهای [۸] تبدیل برنامه از HDL به SystemC به صورت اتوماتیک انجام شود. تعریف هر المان برای قرار گرفتن در کتابخانه ار دو بخش تشکیل می‌شود یک فایل هدر و یک فایل C++. در فایل هدر ورودی و خروجی‌های المان، لیست حساسیت و پروسه C++ تعیین می‌شود در این فایل علاوه بر ورودی‌های معمول گیت یک ورودی از نوع <BOOL> تعریف شده است در صورت یک بودن این ورودی خروجی گیت به صورت تصادفی در مقدار صفر یا یک قرار می‌گیرد (خطای Sa0 و Sa1) البته می‌توان به سادگی با تغییر نوع این ورودی به <sc_logic> گستره تنوع خطاها را افزایش داد اما هدف ما در این مقاله خطای انباشتگی در منطق صفر یا یک می‌باشد چرا که این نوع خطا بسیار شایع‌تر است. در فایل پروسه که نسبت به هر یک از ورودی‌ها و خط خطا حساس می‌باشد در صورت فعال بودن خط خطا (یک منطقی) خروجی گیت به صورت تصادفی یک یا صفر خواهد شد و در غیر این صورت خروجی معرف عملکرد واقعی گیت خواهد بود. نمونه‌ای از فایل هدر و پروسه آن برای گیت AND سه ورودی در شکل ۱ آمده است.

بستر تست: بستر تست وظیفه مدیریت ورودی‌ها، تزریق خطا و آنالیز خروجی‌های را بر عهده دارد. الگوریتم این بخش عبارت است:

۱- مقداردهی اولیه متغیرها (شامل: تعداد دفعات تکرار، تعداد دفعات آشکارسازی خطا، مقادیر اولیه ورودی‌ها و ...)
۲- قرار دادن بردار تصادفی در ورودی مدار تحت تست (ایجاد یک عدد تصادفی و تبدیل آن به لاجیک)
۳- انتخاب تصادفی یکی از المان‌ها و فعال‌سازی خط خطا در آن المان (احتمال انتخاب هر المان متناسب با تعداد ترانزیستورهای آن می‌باشد)
۴- انتظار به اندازه یک سیکل (جهت مشاهده خروجی)
۵- ارزیابی خروجی مقایسه‌کننده
۵-۱- در صورت یک بودن افزایش شمارنده مشاهده خطا
۵-۲- عدم تغییر در شمارنده خطا در صورت صفر بودن
۶- تکرار مراحل قبل به تعداد دفعات تزریق؛ گزارش شمارنده خطا

مدار تحت تست: مدار تحت تست، توصیف ساختاری از مدار مطلوب جهت تست تزریق خطا می‌باشد. المان‌های مورد استفاده در این بخش از کتابخانه المان‌ها جایگذاری می‌شود. ورودی‌های این بخش به‌طور مستقیم از بستر تست اعمال می‌شود علاوه بر آن ورودی‌های مربوط به تزریق خطا نیز از بستر تست کنترل می‌گردد خروجی این بخش در اختیار واحد مقایسه کننده قرار می‌گیرد.

مدار موید: مدار موید از نظر ساختار کاملا مشابه مدار تحت تست می‌باشد. ورودی و خروجی‌های آن مشابه مدار تحت تست، به بستر تست و مقایسه‌کننده وصل‌اند با این تفاوت که ورودی‌های مربوط به خطا همگی دارای منطق صفر هستند و خطایی در این بخش تزریق نمی‌شود نتایج این بخش برای تایید درستی خروجی‌های مدار تحت تست استفاده می‌شود.

مدار مقایسه کننده: بخش مقایسه‌کننده که در سیستم‌های دوپلکس نیز جهت آشکار سازی هم‌زمان خطا استفاده می‌شود به صورت توصیف ساختاری از گیت‌های XOR و OR بر اساس المان‌های کتابخانه المان‌ها پیاده‌سازی شده است. اما چنانچه در شکل۱ نیز نشان داده شده است ورودی‌های مربوط به خطوط خطا در این بخش نیز غیر فعال است دلیل این امر در محاسبات درصد پوشش خطا مشخص شده است. یک بودن خروجی این بخش نشان‌دهنده انتقال خرابی به خروجی (آشکارسازی خطا) و صفر بودن آن به معنای عدم آشکارسازی خطای تزریق شده می‌باشد.

درصد پوشش خطا

برای تعیین درصد پوشش خطا در یک سیستم الکترونیکی دوپلکس به روش تزریق خطا می‌توان خطوط ورودی خطا در مدار موید و مقایسه-کننده را نیز به بستر تست متصل کرده و تزریق خطا در آن بخش‌ها نیز انجام شود اما با فرض n_{IN}: تعداد ورودی‌های مدار ترکیبی، n_M: تعداد گیت‌های مدار تحت تست، n_C: تعداد گیت‌های بخش مقایسه-کننده، این کار زمان لازم برای شبیه‌سازی تزریق خطا را از مرتبه $2^{(n_{in}+n_M)}$ به مرتبه $2^{(n_{in}+2n_M+n_C)}$ افزایش می‌دهد.

با توجه به ساختار بخش مقایسه‌کننده مشخص می‌شود که مسیرها و المان‌های این بخش به گونه‌ای قرار گرفته‌اند که هر خطای واحد در خروجی قابل مشاهده است برای تست این فرضیه کافی است مدار تحت تست مدار مقایسه‌کننده قرار داده شود در این صورت درصد پوشش خطا با تزریق خطا در تمام بخش‌ها ۱۰۰درصد خواهد بود.

در سیستم دوپلکس اگر احتمال رخداد خطا در خود ماژول P_f باشد احتمال رخداد خطا در بخش مقایسه کننده $1\text{-}2P_f$ خواهد بود و این احتمال از رابطه زیر تعیین می‌شود.

$$P_f = \frac{\left\langle \sum_{i=0}^{i=n_M} Ng_i \right\rangle}{2\left\langle \sum_{i=0}^{i=n_M} Ng_i \right\rangle + \sum_{i=0}^{i=n_C} Ng_{ci}} \qquad (۱)$$

رابطه (۱) احتمال خرابی را بر مبنای تعداد ترانزیستورهای هر بخش تشکیل شده است این تعداد به طور تقریبی متناسب با تعداد ورودی‌های آن گیت می‌باشد. در این رابطه Ng_i تعداد ورودی‌های گیت iام در مدار تحت تست و Ngc_i تعداد ورودی‌های گیت iام در بخش مقایسه‌کننده می‌باشد.

در صورتی که عدد گزارش شده از الگوریتم بستر تست نسبت به تعداد دفعات تزریق μ باشد در این صورت درصد پوشش خطا برای سیستم دوپلکس از رابطه زیر تعیین می‌شود:

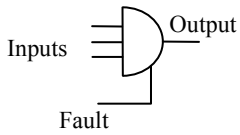
$$Fault\_Coverage = (2P_f)\mu + (1-2P_f) \qquad (۲)$$

برای ارزیابی کارایی روش ارائه شده، مدارهای محک استاندارد ISCAS به کار برده شده‌اند[۹] نتایج در جدول ۱ آمده است.

نتیجه‌گیری

در این مقاله مکانیزمی مبتنی بر ساختارهای تغییرپذیر در محیط SystemC ارائه شد. این تکنیک با توجه به ادغام بستر تست در مدار تحت تست از یک‌سو و استفاده از ساختار ماژولار بودن توسعه از سوی دیگر نسبت به روش‌های مشابه راحتر و سریعتر می‌باشد. با استفاده از این مکانیزم قابلیت اطمینان چند مدار محک ارزیابی و گزارش گردید.

شکل‌ها و نمودارها

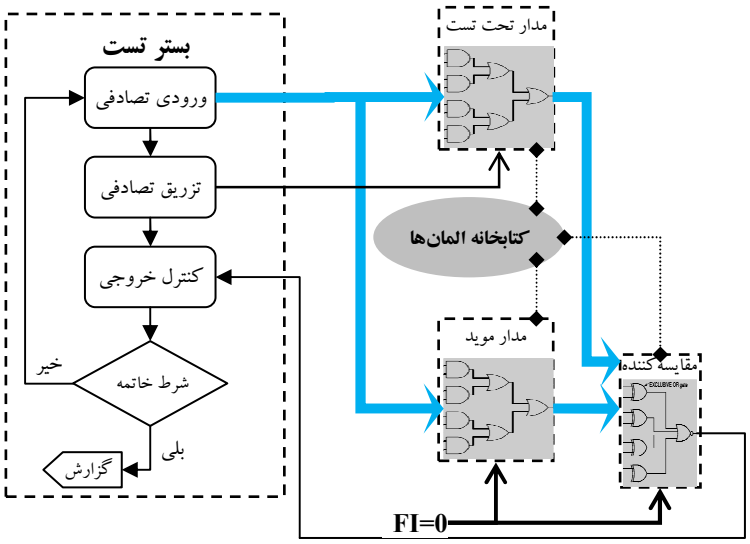



```
SC_MODULE(AND3)
{
public:
    /*Ports*/
sc_in<sc_logic>    in1;
sc_in<sc_logic>    in2;
sc_in<sc_logic>    in3;
sc_in<bool >       FI; // Fault Injection
sc_out<sc_logic>   dout;
private:
    /*Processes*/
    void run();
public:
    /*Constructor*/
    SC_CTOR(AND3_Pro)
    {
          SC_METHOD(run);
          sensitive << clock_in.pos();
          sensitive << set;
          sensitive << reset;
    };
};
```

```
#include <systemc.h>
#include "AND3.sc.h"

void AND3_Pro::run()
{
    /*
     * Reset cycle, before the first
     */
rand_num=rand()%2;
if (FI)
  switch(rand_num)
        {
   case 0: dout =sc_logic('0');break;
   case 1: dout =sc_logic('1');break;
         }
else
    dout=in1&in2&in3;
};
```

شکل ۱: فایل هدر و پروسه C++ مربوطه برای گیت AND سه ورودی

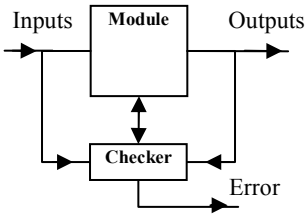


شکل ۲: ساختار ارائه شده جهت تزریق خطا در محیط SystemC

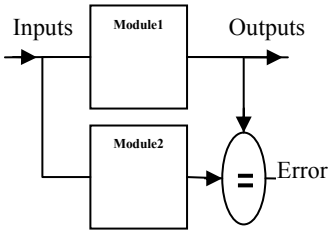
جدول ۱:نتایج شبیه‌سازی تزریق خطا در مدارهای مختلف

Circuit	*inputs*	*output*	*gates*	*gates in duplex*	*Fault Coverage*	*Simulation Time of Proposed Method*[+]	*Simulation Time of Duplex Method*[+]
C17	5	2	6	13	75.01%	***2.2 E-1***	***2.8 E+1***
74182 : 4-bit carry-lookahead generator	9	4	19	45	81.61%	***2.9 E+4***	***1.9 E+12***
74283 : 4-bit adder	9	5	36	82	78.80%	***3.8 E+9***	***2.7 E+23***
74L85 : 4-bit magnitude comparator	11	3	33	72	50.12%	***1.9 E+9***	***1.0 E+21***
4-bit fast carry skip adder	9	4	28	63	78.22%	***4.1 E+18***	***5.1 E+17***
8-bit fast carry skip adder	17	8	56	127	80.92%	***1.0 E+18***	***2.4 E+39***

[+] تخمین زمان بر حسب ثانیه گزارش شده است.



شکل ۳: ساختار مدارهای خودرس جهت تشخیص هم‌زمان خطا



شکل ٤: ساختار دوپلکس جهت تشخیص هم‌زمان خطا

مراجع

1. Reinaldo Perez; "Methods for Spacecraft Avionics Protection against Space Radiation in the Form of Single-Event Transients," IEEE TRANSACTIONS ON ELECTROMAGNETIC COMPATIBILITY, VOL. 50, NO. 3, AUGUST 2008.

2. Dominik, L.; "System mitigation techniques for single event effects," Digital Avionics Systems Conference, 2008, Page(s): 5.C.2-1 - 5.C.2-12.

3. Shafik, R.A.; Rosinger, P.; Al-Hashimi, B.M.; "SystemC-Based Minimum Intrusive Fault Injection Technique with Improved Fault Representation," On-Line Testing Symposium, 2008.

4. Dongwoo Lee; Jongwhoa Na; "A Novel Simulation Fault Injection Method for Dependability Analysis," Design & Test of Computers, IEEE Journals; 2009, Page(s): 50 – 61.

5. Jenn, E. ; Arlat, J. ; Rimen, M. ; Ohlsson, J. ; Karlsson, J. ; "Fault Injection into VHDL Models: The MEFISTO Tool," Twenty-Fourth International Symposium on Fault-Tolerant Computing, 1994.

6. S.J. Seyyed Mahdavi ; K. Mohammadi ; "Improved single-pass approach for reliability analysis of digital combinational circuits," Microelectronics Reliability Volume 51, Issue 2, February 2011, Pages 477-484.

7. J. Gracia; J.C. Baraza; D. Gil; P.J. Gil; "Comparison and Application of different VHDL-Based Fault Injection Techniques," Proceedings of the 2001 IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems.

8. http://www.ht-lab.com/freeutils/vh2sc/vh2sc.html

9. http://www.eecs.umich.edu/~jhayes/iscas/