بسم الله الرحمن الرحیم

دانشگاه تربیت دبیر شهید رجائی

مبانی مدل‌سازی SoC با

SystemC
و
SystemC-AMS

به همراه

تألیف:

دکتر پرویز امیری

عضو هیأت علمی دانشگاه تربیت دبیر شهید رجائی

مهندس منا خزایی‌زاده

سر شناسنامه	:	**امیری، پرویز، ۱۳۴۸-**
عنوان و نام پدید آور	:	**مبانی مدل‌سازی SoC با SystemC و SystemC-AMS/ تألیف پرویز امیری، منا خزایی‌زاده.**
مشخصات نشر	:	**تهران: دانشگاه تربیت دبیر شهید رجائی، ۱۳۹۴.**
مشخصات ظاهری	:	**۳۹۰ ص.: مصور (بخشی رنگی) + یک لوح فشرده.**
شابک	:	**۹۷۸-۶۰۰-۶۵۹۴-۴۷-۷**
وضعیت فهرست نویسی	:	**فیپا**
یادداشت	:	**واژه‌نامه.**
یادداشت	:	**کتابنامه: ص. ۳۸۱.**
موضوع	:	**سیستم‌های روی تراشه**
موضوع	:	**سیستم‌های کامپیوتری درونه‌ای**
موضوع	:	**سی ++ (زبان برنامه نویسی کامپیوتر)**
شناسه افزوده	:	**خزایی‌زاده، منا، ۱۳۶۶-**
شناسه افزوده	:	**دانشگاه تربیت دبیر شهید رجائی**
رده بندی کنگره	:	**TK۷۸۹۵ /۱۳۹۴ ۸الف۹س**
رده بندی دیویی	:	**۰۰۴/۶۵**
شماره کتابشناسی ملی	:	**۴۰۶۱۸۵۳**

دانشگاه تربیت دبیر شهید رجائی

عنوان	:	مبانی مدل‌سازی SoC با SystemC و SystemC-AMS
تألیف	:	دکتر پرویز امیری، عضو هیأت علمی دانشگاه تربیت دبیر شهید رجائی، مهندس منا خزایی‌زاده
ویراستار علمی	:	دکتر زین العابدین موسوی
ویراستار ادبی	:	دکتر یداله بهمنی
نوبت چاپ	:	اول- پاییز ۱۳۹۴
انتشارات	:	دانشگاه تربیت دبیر شهید رجائی
لیتوگرافی	:	فرانقش
چاپ	:	فردوس
طراح جلد	:	مهندس محمد معتمدی‌نژاد
ناظر چاپ	:	مهندس محمد معتمدی نژاد
کارشناس چاپ و صفحه‌آرا	:	نیره فیروزی
کارشناسان	:	طاهره کیاء/ علی رضایی اهوانوئی
شمارگان	:	۱۰۰۰ جلد
قیمت	:	۲۰،۰۰۰ تومان
شابک	:	۹۷۸-۶۰۰-۶۵۹۴-۴۷-۷ ISBN: 978-600-6594-47-7

نشانی: تهران، لویزان – کد پستی ۱۶۷۸۸-۱۵۸۱۱ – صندوق پستی ۱۶۳ – ۱۶۷۸۵ – تلفن: (۲۶۳۲) ۹ – ۲۲۹۷۰۰۶۰، ۲۲۹۷۰۰۷۰، تلفکس: ۲۲۹۷۰۰۴۲، پست الکترونیکی: Publish@srttu.edu، وب سایت: http://Publish.srttu.edu

تقدیم به مادران و پدرانمان

به پاس محبت‌ها و حمایت های

بی دریغشان که هرگز

فروکش نمی‌کند

پیش‌گفتار

پیشرفت‌های فن‌آوری توانایی مجتمع‌سازی عملکردهای بیشتر و بیشتر را بر روی یک تراشه فراهم می‌سازد و صنعت الکترونیک را به سوی یک الگوی طراحی جدید به نام سیستم بر روی تراشه (SoC) سوق می‌دهد. در طراحی‌های SoC تمام قابلیت‌های یک سیستم کامل با تمرکز بیشتر بر حجم نرم‌افزار در یک تراشه‌ی سیلیکونی یکتا قرار داده می‌شود، که این کار منجر به افزایش عملکرد،کاهش توان مصرفی،کاهش هزینه و کاهش اندازه‌ی قطعات می‌شود.

بنابراین یکی از بزرگ‌ترین چالش‌های جهان امروز، طراحی و ساخت SoC در حجم، وزن و زمان کمتر و دقت بیشتر است. در این راستا، رقابت صاحبان صنایع در طراحی سیستم‌های گوناگون بهینه منجر به پیدایش احساس نیاز به وجود یک محیط شبیه‌سازی یک‌پارچه برای مدل‌سازی سیستم‌های پیچیده و کامل شامل الکترونیک دیجیتال و آنالوگ، میکرو ویو RF و کاربردهای نرم‌افزاری و غیره بر روی یک فیزیک مدار مجتمع، شده است.

یک زبان مدل‌سازی یکتا که بتواند برای توصیف یک سیستم در تمام سطوح انتزاع به‌کار برده شود، به طور چشم‌گیری زمان و زحمت طراحی را کاهش می‌دهد. نیاز به بازنویسی مدل در طول روند طراحی، حذف می‌شود. همان تست بنچ‌هایی که در مراحل اولیه برای ارزیابی طرح، نوشته شدند، می‌توانند در سایر سطوح انتزاع نیز استفاده شوند، که این کار منجر به کاهش هزینه و زمان ساخت می‌شود. استفاده از یک زبان همچنین موجب حصول اطمینان از یکدستی و عاری از خطا بودن مدل‌ها در طول تمام سطوح انتزاع و در طی انتقال از سطوح بالاتر به سطوح پایین‌تر می‌شود. بنابراین، نیاز به یک پلت فرم/زبان مدل‌سازی وجود دارد که بتواند

مدل‌سازی رفتاری سطح بالا تا انتزاع سطح پایین یا مدل‌های RTL را به خوبی پشتیبانی کند[۱].

این پلت‌فرم مدل‌سازی مطلوب، همچنین باید سنتز مدل‌ها را درون اجزای سخت‌افزار یا نرم‌افزار پشتیبانی کند. یک پلت‌فرم مدل‌سازی SoC باید ابزارهای سنتز ذاتی داشته باشد، چراکه تبدیل مدل‌ها از یک پلت‌فرم به دیگری، فرآیند پرهزینه‌ای است. در واقع همان‌طور که ابزارهایی برای سنتز سخت‌افزار وجود دارد، نیاز به ابزارهایی برای سنتز نرم‌افزار نیز حس می‌شود. سال‌ها است که ابزارهایی برای تبدیل مدل‌های انتزاع سطح-بالا برای مدار سخت‌افزاری عملی تعریف شده است، اما ابزارهای مشابه برای سنتز نرم‌افزار وجود ندارد. برای مدیریت پیچیدگی همواره در حال رشد سیستم‌ها، اتوماتیک‌سازی مراحل سنتز نرم‌افزار دیگر یک انتخاب نخواهد بود، بلکه یک نیاز است. ممکن است این سؤال مطرح شود که چرا به مدل-سازی سخت‌افزار در سطح RT نیاز داریم، در حالی که پاسخ‌گویی سطح سیستم سریع‌تر است. دلیل اصلی که استفاده از سطوح گوناگون در طول فرآیند طراحی تا ساخت، این است که سیستم‌های بزرگ معمولاً از اجزای گوناگونی تشکیل شده‌اند که دارای شرایط تأثیرگذار متغیری بر کارایی کل سیستم می‌باشند از قبیل کنترل مساحت، سادگی، انعطاف‌پذیری پیاده-سازی و هزینه. برای چنین سیستم‌هایی، به محدوده‌ی بزرگی از سطوح انتزاع برای مدیریت مقتضیات و موازنه‌های آنها نیاز داریم[۲]. به عنوان مثال، در کاربرد مخابرات فرکانس رادیویی، کاهش بخش‌های RF و افزایش توانایی‌های بیشتر در بخش نرم‌افزار، انعطاف‌پذیری طرح را برای تغییر، بالا می‌برد.

زبان‌های توصیف سخت‌افزاری، یا HDLها، زبان‌هایی هستند که برای طراحی سخت‌افزار استفاده می‌شوند. HDLها امکان شبیه‌سازی طرح در سیکل طراحی به منظور تصحیح خطاها و یا آزمایش با معماری‌های مختلف را فراهم می‌سازند. طرح‌هایی که در HDL توصیف می‌شوند مستقل از تکنولوژی بوده و طراحی و اشکال‌زدایی آنها آسان است و معمولاً بیشتر از شماتیک‌ها به خصوص برای مدارهای بزرگ و پیچیده، قابل خواندن هستند.

از ویژگی‌های اصلی زبان توصیف سخت‌افزار، توانایی آن در توصیف عملکرد یک قطعه از سخت‌افزار، مستقل از پیاده‌سازی است. هدف از پیشرفت در صنعت HDL، دستیابی به یک زبان یک‌پارچه برای توصیف عملکرد طرح تا پیاده‌سازی نهایی است. این کار منجر به انجام تمام فرآیند طراحی در یک زبان مشخص و در نتیجه، یک ارائه‌ی منفرد از طراحی می‌شود. تلاش‌ها و اقدامات بسیاری در این زمینه صورت گرفته است؛ اما این زبان‌ها دارای

محدودیت‌هایی از جمله عدم پشتیبانی کامل از مدارهای آنالوگ و همچنین مدل‌سازی و سنتز نرم‌افزار هستند [۳].

بنابراین نیاز به زبان‌های جامع‌تری که بتوانند از تمام سطوح انتزاع و مدل‌سازی نرم‌افزار و حتی آنالوگ پشتیبانی کنند، موجب ظهور نسل جدیدی از زبان‌ها مانند Verilog-AMS و VHDL-AMS و System Verilog و SystemC و SystemC-AMS شد، که از میان اینها تنها SystemC قادر به مدل‌سازی همزمان نرم‌افزار نیز می‌باشد.

SystemC که بر C++ بنیان شده است، انعطاف‌پذیری را برای مدل‌سازی طرح در سطوح گوناگون انتزاع فراهم می‌کند. همچنین سنتز نرم‌افزاری توسط C++ در این زبان فراهم می‌باشد. این زبان که مدل‌سازی از سطوح بالا در قالب رفتاری تا سطح RTL را پشتیبانی می‌کند، کُد باز بوده و یادگیری و استفاده‌ی آن بسیار آسان است. مهم‌ترین مزیت این زبان این است که می‌تواند به عنوان یک زبان مشترک میان طراحان سطح سیستم، مهندسان نرم‌افزار و طراحان سخت‌افزار استفاده شود. نسخه‌ی ۱,۴ آن هم‌اکنون در محیط‌هایی مانند VIVADO Xilinx قابل سنتز می‌باشد. مؤسسه‌ی SystemC متعلق به یک سازمان مستقل متشکل از محدوده‌ی وسیعی از شرکت‌ها، دانشگاه‌ها و افراد خبره به پشتیبانی و پیشبرد SystemC به عنوان استاندارد رایگان و در دسترس برای طراحی سطح سیستم می‌باشد [۴]. OSCI[1] یک مجموعه از رابط‌های TLM است که چگونگی ارتباط مدل‌ها، و مجموعه‌ای از سطوح انتزاع را برای مدل‌سازی طرح‌ها تعیین می‌کند. هدف این مجموعه پشتیبانی کامل از سطوح مختلف نمایش یک سیستم کامل و پیچیده است که با شرح جزئیات در محدوده‌ای از توصیف عملکردی تا مدل سخت‌افزاری قابل سنتز هستند[۵].

SystemC-AMS، یک نسخه‌ی تکمیلی از SystemC است که در سال ۲۰۰۶ ارائه شده و با تکیه بر قابلیت‌های C++ و SystemC توابعی را برای مدل‌سازی رفتار آنالوگ و سیگنال مختلط و فرکانس بالا در یک سیستم ارائه می‌کند. این کتابخانه قابلیت مدل‌سازی سیستم‌ها را در هر دو بخش نرم‌افزار و سخت‌افزار به همراه اثرات غیرخطی در بیشتر سطوح انتزاع دارد.

SystemC و SystemC-AMS دارای توانایی‌های بسیار خوبی هستند که با توجه به کُد باز بودن، هنوز در حال پیشرفت و توسعه هستند. از سوی دیگر با توجه به یادگیری آسان و در دسترس بودن تمام ابزارهای مورد نیاز برای کار با این دو زبان، این زبان می‌تواند در صنعت و دانشگاه مورد استفاده‌ی مهندسان از تازه‌کارها تا افراد مجرب و خبره قرار گیرد[۶] تا [۱۲].

1- Open System C Initiative

در این کتاب به معرفی مقدماتی SystemC و SystemC-AMS پرداخته شده است. هر چند فرض بر این است که خواننده دانش مقدماتی و پایه در مورد زبان برنامه‌نویسی C++ دارد چرا که برای استفاده از این دو زبان باید با اصول C++ آشنا باشیم، اما اشاراتی به توابع ضروری C++ در استفاده از این دو زبان شده است. در فصل اول ابتدا با SoC و سطوح مدل‌سازی آنها و زبان‌های مختلف و کارایی آنها در مدل‌سازی SoCها آشنا می‌شویم و سپس نیاز به استفاده از این دو زبان در مدل‌سازی SoC مطرح می‌شود. در فصل دوم، به معرفی کامل SystemC و نحوه‌ی مدل‌سازی با استفاده از این کتابخانه و ساختار کلی و توابع آن پرداخته می‌شود. در فصل سوم، سه مدل محاسباتی اصلی در SystemC-AMS به طور کامل به همراه مثال‌های کاربردی و متودولوژی‌های مدل‌سازی و ذخیره‌ی نتایج و آنالیز حوزه‌ی زمان و فرکانس توضیح داده می‌شود. در فصل چهار، یک جمع‌بندی از کلیه‌ی توابع ارائه می‌شود که در دسترسی سریع به مدل‌ها به کاربران بسیار کمک می‌کند. در نهایت در دو فصل پایانی این کتاب آموزش گام به گام روش نصب و استفاده از SystemC و SystemC-AMS و مثال‌های آموزشی کامل به همراه کدهای کامل و نتایج آنها آورده شده است.

نرم‌افزارهای لازم و کتابخانه‌های هر دو پلت‌فرم در CD ضمیمه این کتاب به همراه کدهای کامل مثال‌های فصل ۶ آورده شده است. مسلماً این مجموعه از نقص و کاستی مبرا نیست لذا نظر اصلاحی صاحب‌نظران می‌تواند در رفع کاستی‌های آن یاری‌گر مؤلفان باشد. امید است این مجموعه بتواند در رفع نیاز علاقه‌مندان در این زمینه نقش کوچکی داشته باشد.

پرویز امیری
منا خزایی‌زاده
پاییز ۱۳۹۴

فهرست مطالب

عنوان	صفحه
فصل اول: آشنایی با SoC، زبان‌ها و سطوح مدل‌سازی	**۱**
۱-۱. مقدمه‌ای بر SoC	۳
۱-۲. سطوح مدل‌سازی	۱۰
۱-۲-۱. سطح انتقال ثبات	۱۱
۱-۲-۲. سطح رفتاری	۱۲
۱-۲-۳. سطح تراکنش TLM	۱۳
۱-۳. زبان‌های طراحی و مدل‌سازی آنالوگ و سیگنال‌مختلط	۱۷
۱-۳-۱. VHDL و VHDL-AMS	۱۷
۱-۳-۲ . Verilog و توسعه‌های آن	۱۸
۱-۳-۳. MATLAB	۲۰
۱-۳-۴ . SPICE	۲۱
۱-۴. ضرورت استفاده از SystemC	۲۱
۱-۵ . معرفی SystemC	۲۳
۱-۶. طراحی در سطوح مختلف انتزاع در SystemC	۲۵
۱-۶-۱.مدل الگوریتمی	۲۵

۱-۶-۲. مدل سطح رفتاری ۲۶
۱-۶-۳. مدل سطح تراکنش TLM ۲۶
۱-۶-۳-۱.TLM مدل عملکردی فاقدزمان‌بندی ۲۶
۱-۶-۳-۲. TLM مدل عملکردی زمان‌بندی‌شده ۲۷
۱-۶-۴. مدل سطح انتقال ثبات ۲۷
۱-۷. SystemC_AMS چیست؟ ۲۷
۱-۷-۱. مزایا و قابلیت‌ها ۲۹
۱-۸. چشم‌انداز آینده ۳۶

فصل دوم: ساختار اساسی برنامه‌نویسی با SystemC **۳۷**
۲-۱.کار با SystemC ۳۹
۲-۲. معرفی بخش‌های کلی ساختار ۴۳
۲-۲-۱. ساختار فایل ۴۵
۲-۲-۲. ماژول ۴۸
۲-۲-۳. SC_CTOR ۵۱
۲-۲-۴. نگه دارنده‌های مقدار ۵۲
۲-۲-۵. خواندن و نوشتن پورت‌ها ۵۴
۲-۲-۶. عمل‌گرها ۵۵
۲-۲-۷. دستورهای شرطی ۶۱
۲-۲-۸. حلقه ۶۶
۲-۲-۹. انتخاب بیت و محدوده ۶۸
۲-۲-۱۰. Process ۷۱
۲-۳. مدل‌سازی منطق ترتیبی و ترکیبی ۸۶
۲-۳-۱. مدل‌سازی منطق همزمانی ۸۶
۲-۳-۲. processهای چندگانه ۸۷
۲-۴. تولید، ذخیره و نمایش نتایج ۸۸

۲-۵. تست‌بنچ ۸۹

۲-۵-۱. sc_clock ۹۰

۲-۵-۲. sc_trace ۹۱

۲-۵-۳. sc_start ۹۱

۲-۵-۴. sc_stop ۹۱

۲-۵-۵. sc_time_stamp ۹۲

۲-۵-۶. sc_simulation_time ۹۲

۲-۵-۷. sc_time ۹۲

۲-۶. انواع داده ۹۷

۲-۶-۱. نوع داده‌ی بیت ۹۸

۲-۶-۲. نوع داده‌ی بیت با طول دلخواه ۹۹

۲-۶-۳. نوع داده‌ی منطقی ۱۰۲

۲-۶-۴. نوع داده‌ی عدد صحیح علامت‌دار ۱۰۳

۲-۶-۵. نوع داده‌ی resolved ۱۰۵

۲-۶-۶. نوع داده‌ی تعریف شده توسط کاربر ۱۰۷

فصل سوم: SystemC_AMS **۱۰۹**

۳-۱. مقدمه ۱۱۱

۳-۲. TDF ۱۱۲

۳-۲-۱. ماژول‌های TDF ۱۱۴

۳-۲-۲. خصوصیات ماژول و درگاه TDF ۱۱۶

۳-۲-۳.توپولوژی‌های مدل TDF ۱۱۸

۳-۲-۴. تنظیم خصوصیت‌های درگاه ۱۲۲

۳-۲-۵. گام‌زمانی ۱۲۳

۳-۲-۶. خوشه‌ها یا برنامه‌های چندگانه ۱۲۷

۳-۲-۷. رفتار پردازش سیگنال مدل‌های TDF	۱۲۸
۳-۲-۸. درگاه‌های TDF	۱۳۲
۳-۲-۹. سیگنال‌های TDF	۱۳۷
۳-۲-۱۰. مدل‌سازی زمان‌گسسته و رفتار زمان‌پیوسته	۱۳۸
۳-۲-۱۱. رفتار چند نرخی	۱۵۲
۳-۲-۱۲. معرفی تأخیرها	۱۵۵
۳-۲-۱۳. تعامل میان TDF و حوزه‌ی زمان‌گسسته	۱۵۶
۳-۲-۱۴. روند اجرای TDF	۱۶۱
۳-۳. LSF	۱۶۲
۳-۳-۱. اصول مدل‌سازی LSF	۱۶۲
۳-۳-۲. تنظیم سیستم معادله‌ای LSF	۱۶۳
۳-۳-۳. انتشار و انتساب گام‌زمانی	۱۶۴
۳-۳-۴. سازنده‌های زبان	۱۶۵
۳-۳-۵. مدل‌سازی رفتار زمان‌پیوسته	۱۶۹
۳-۳-۶. اجرای LSF	۱۷۹
۳-۳-۷. مثال کاربردی	۱۸۰
۳-۴. ELN	۱۸۲
۳-۴-۱. راه‌اندازی سیستم معادلاتی	۱۸۳
۳-۴-۲. انتساب انتشار و گام‌زمانی	۱۸۴
۳-۴-۳. ماژول‌های ELN	۱۸۴
۳-۴-۴. گام زمانی ماژول	۱۸۶
۳-۴-۵. ترمینال‌های ELN	۱۸۷
۳-۴-۶. گره‌های ELN	۱۸۸
۳-۴-۷. مدل‌سازی رفتار زمان‌پیوسته	۱۸۹
۳-۴-۸. روند اجرایی ELN	۱۹۷

۳-۵. تحلیل حوزه‌ی فرکانس ۱۹۸
۳-۵-۱. آنالیز سیگنال‌کوچک حوزه‌ی فرکانس ۱۹۸
۳-۵-۲. توصیف سیگنال‌کوچک حوزه‌ی فرکانس در ماژول‌های TDF ۱۹۹
۳-۶. انجام شبیه‌سازی و ایجاد نتایج ۲۰۰
۳-۶-۱. کنترل شبیه‌سازی در حوزه‌ی زمان ۲۰۰
۳-۶-۲. کنترل شبیه‌سازی سیگنال‌کوچک حوزه‌ی فرکانس ۲۰۲
۳-۶-۳. فرمت‌ها و فایل‌های trace ۲۰۳
۳-۶-۴. کنترل فایل ترسیم ۲۰۶
۳-۶-۵. نوشتن نکته یا نشانه در فایل ترسیم ۲۰۹

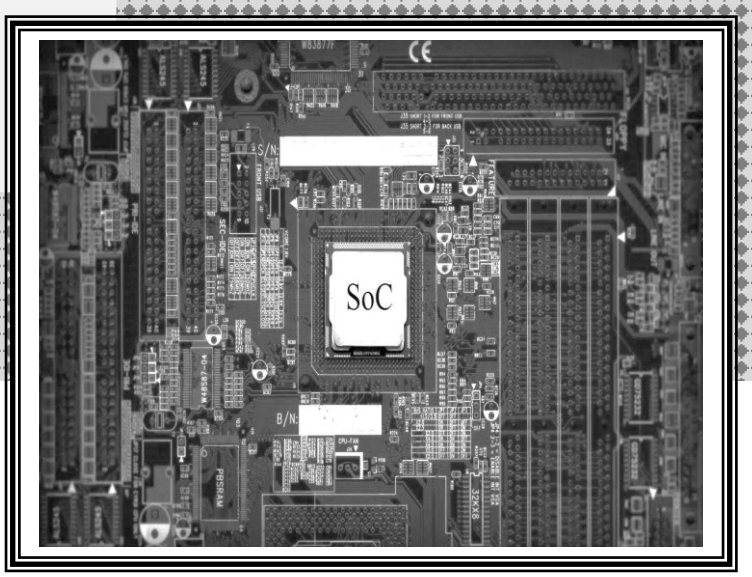
فصل چهارم: آموزش سریع استفاده از ماژول‌های پیش‌فرض ۲۱۱
۴-۱. ماژول TDF ۲۱۳
۴-۱-۱. سیگنال TDF ۲۱۴
۴-۱-۲. توابع انتقال لاپلاس ۲۱۴
۴-۱-۳. sca_tdf::sca_ltf_nd ۲۱۴
۴-۱-۴. sca_tdf::sca_ltf_zp ۲۱۵
۴-۱-۵. sca_tdf::sca_ss ۲۱۵
۴-۲. ماژول LSF ۲۱۷
۴-۲-۱. sca_lsf::sca_add ۲۱۷
۴-۲-۲. sca_lsf::sca_sub ۲۱۸
۴-۲-۳. sca_lsf::sca_gain ۲۱۹
۴-۲-۴. sca_lsf::sca_dot ۲۲۰
۴-۲-۵. sca_lsf::sca_integ ۲۲۱
۴-۲-۶. sca_lsf::sca_delay ۲۲۲
۴-۲-۷. sca_lsf::sca_source ۲۲۳

۴-۲-۸. sca_lsf::sca_ltf_nd ۲۲۵
۴-۲-۹. sca_lsf::sca_ltf_zp ۲۲۶
۴-۲-۱۰. sca_lsf::sca_ss ۲۲۷
۴-۲-۱۱. sca_lsf::sca_tdf_gain ۲۲۹
۴-۲-۱۲. sca_lsf::sca_tdf_source ۲۳۰
۴-۲-۱۳. sca_lsf::sca_tdf_sink ۲۳۰
۴-۲-۱۴. sca_lsf::sca_tdf ۲۳۱
۴-۲-۱۵. sca_lsf::sca_tdf_demux ۲۳۲
۴-۲-۱۶. sca_lsf::sca_de_gain ۲۳۳
۴-۲-۱۷. sca_lsf::sca_de_source ۲۳۴
۴-۲-۱۸. sca_lsf::sca_de_sink ۲۳۵
۴-۲-۱۹. sca_lsf::sca_de_mux ۲۳۶
۴-۲-۲۰. sca_lsf::sca_de_demux ۲۳۷
۴-۲-۲۱. sca_lsf::sca_de_gain ۲۳۸
۴-۲-۲۲. sca_lsf::sca_de_source ۲۳۹
۴-۲-۲۳. sca_lsf::sca_de_sink ۲۴۰
۴-۲-۲۴. sca_lsf::sca_de_mux ۲۴۱
۴-۲-۲۵. sca_lsf::sca_de_demux ۲۴۲
۴-۳. ماژول‌های ELN ۲۴۳
۴-۳-۱. sca_eln::sca_r ۲۴۳
۴-۳-۲. sca_eln::sca_c ۲۴۴
۴-۳-۳. sca_eln::sca_l ۲۴۵
۴-۳-۴. sca_eln::sca_vcvs ۲۴۶
۴-۳-۵. sca_eln::sca_vccs ۲۴۷
۴-۳-۶. sca_eln::sca_ccvs ۲۴۸
۴-۳-۷. sca_eln::sca_cccs ۲۴۹
۴-۳-۸. sca_eln::sca_nullor ۲۵۰
۴-۳-۹. sca_eln::sca_gyrator ۲۵۱

۴-۳-۱۰. sca_eln::sca_ideal_transformer ۲۵۲
۴-۳-۱۱. sca_eln::sca_transmission_line ۲۵۳
۴-۳-۱۲. sca_eln::sca_vsource ۲۵۴
۴-۳-۱۳. sca_eln::sca_isource ۲۵۶
۴-۳-۱۴. sca_eln::sca_tdf_r ۲۵۸
۴-۳-۱۵. sca_eln::sca_tdf_c ۲۵۹
۴-۳-۱۶. sca_eln::sca_tdf_l ۲۶۱
۴-۳-۱۷. sca_eln::sca_tdf_rswitch ۲۶۲
۴-۳-۱۸. sca_eln::sca_tdf_vsource ۲۶۳
۴-۳-۱۹. sca_eln::sca_tdf_isource ۲۶۴
۴-۳-۲۰. sca_eln::sca_tdf_vsink ۲۶۵
۴-۳-۲۱. sca_eln::sca_tdf_isink ۲۶۶
۴-۳-۲۲. sca_eln::sca_de_r ۲۶۷
۴-۳-۲۳. sca_eln::sca_de_c ۲۶۸
۴-۳-۲۴. sca_eln::sca_de_l ۲۶۹
۴-۳-۲۵. sca_eln::sca_de_rswitch ۲۷۰
۴-۳-۲۶. sca_eln::sca_de_vsource ۲۷۲
۴-۳-۲۷. sca_eln::sca_de_isource ۲۷۳
۴-۳-۲۸. sca_eln::sca_de_vsink ۲۷۴
۴-۳-۲۹. sca_eln::sca_de_isink ۲۷۵

فصل پنجم: راهنمایی نصب و استفاده از SystemC و SystemC_AMS ۲۷۷
۵-۱. نصب visual studio۲۰۱۰ ۲۷۹
۵-۲. نصب SystemC ۲۸۳
۵-۳. نصب SystemC-AMS ۲۸۹
۵-۴. ایجاد برنامه با SystemC و Systemc-AMS ۲۹۱
۵-۴-۱. ایجاد یک پروژه‌ی جدید ۲۹۱
۵-۴-۲. افزودن کتابخانه‌های SystemC و Systemc-AMS به برنامه ۲۹۶

فصل ششم: نمونه‌هایی از مدل‌های پیاده‌سازی شده با SystemC-AMS ۳۰۷
۶-۱. تمام‌جمع‌کننده ۳۰۹
۶-۲. تولید شکل موج سینوسی ۳۱۷
۶-۲. فیلتر پائین‌گذر (انتگرال‌گیر) ۳۱۹
۶-۴. فیلتر بالاگذر (مشتق‌گیر) ۳۲۱
۶-۵. مبدل آنالوگ به دیجیتال ۳۲۴
۶-۶. حلقه‌ی قفل فاز ۳۲۷
۶-۷. حلقه‌ی قفل فاز پمپ‌بار ۳۳۶
۶-۸. پیاده‌سازی ترانزیستور MOS ۳۴۵
۶-۹. گیرنده- فرستنده‌ی QPSK ۳۴۶
۶-۹-۱. مشخصه‌های اصلی فرستنده- گیرنده QPSK ۳۴۶
۷-۱۰. مدل‌سازی کانال بی‌سیم ۳۷۱
واژه‌نامه ۳۷۵
منابع ۳۸۱



فصـــل اول

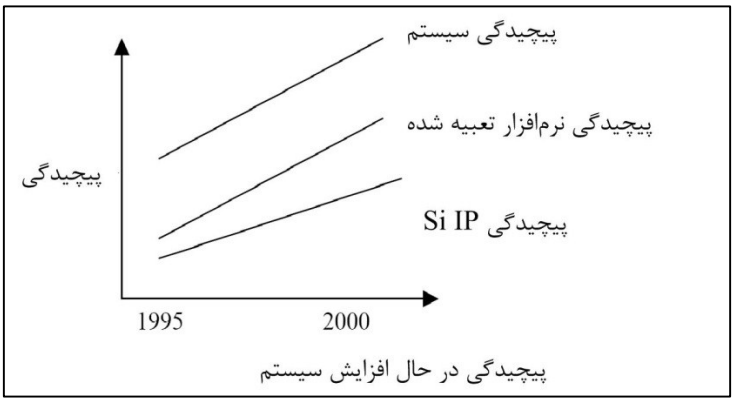
آشنایی با SoC، زبان‌ها و سطوح مدل‌سازی

۱-۱. مقدمه‌ای بر SoC

در سال‌های اخیر پیشرفت‌های چشم‌گیری در تکنولوژی صنعت نیمه‌هادی رخ داده است. پیشرفت‌های ادامه‌دار در تکنولوژی ساخت IC و علم مواد، پیشرفت تکنولوژی را با قانون Moore ممکن می‌سازند[۷]. تعداد ترانزیستورهای ممکن بر روی یک تراشه و فرکانس ساعت هر ۱۸ماه دوبرابر می‌شود. این وضعیت، طراحی سیستم‌های پیچیده را روی یک تراشه ممکن ساخته است و منجر به معماری‌ها و الگوهای طراحی جدید شده است.

در گذشته، سیستم‌ها با استفاده از اجزای مجزا مانند ریزپردازنده، حافظه و اجزای آنالوگ ساخته می‌شدند. این سیستم‌ها از نظر پیچیدگی، عملکرد، سرعت و هزینه کیفیت مطلوبی ندارند. برای افزایش عملکرد تا حد ممکن با اجزای مجزا، باید عملکرد را درون یک تراشه مجتمع کرد. نیاز به مجتمع‌سازی عملکرد، منجر به ظهور طراحی‌های VLSI شد. یک تراشه‌ی VLSI معمولاً یک زیر- سیستم کامل یا یک بخش بزرگ از توابع عملکردی مورد نیاز را پیاده-سازی می‌کند. امروزه یک سیستم ساده شامل هسته‌های گوناگون VLSI، ریزپردازنده و نرم-افزار تعبیه‌شده می‌باشد که در پردازنده‌ها اجرا می‌شود. پیچیدگی سیستم نهایی، اکنون به دلیل پیچیدگی در هسته‌های سیلیکون و نرم‌افزار تعبیه شده می‌باشد. شکل۱-۱ رشد پیچیدگی سیستم‌ها را با زمان نشان می‌دهد.

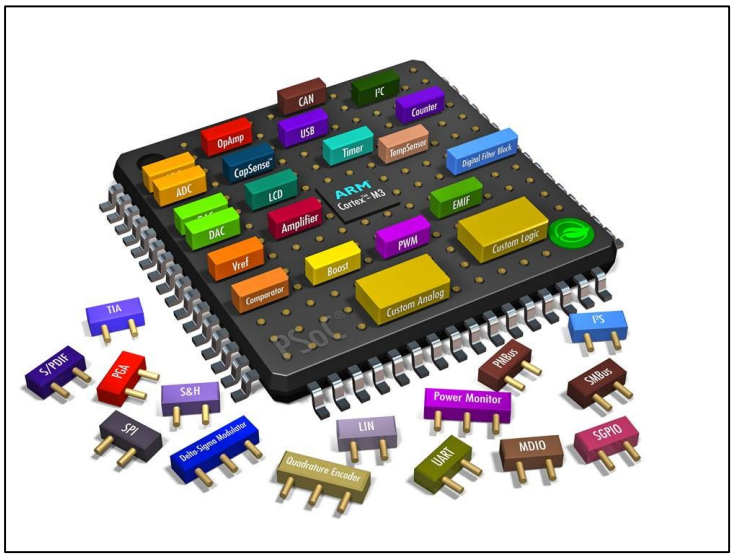
افزایش تقاضا برای عملکردهای هرچه بیشتر درون یک تراشه، منجر به طراحی‌های بزرگ و پیچیده شده است و طراحی‌های سیستم بر اساس تراشه‌های VLSI را با محدودیت مواجه ساخته است. بنابراین روش‌های طراحی قبلی، دیگر نمی‌تواند از عهده‌ی پیچیدگی افزایش یافته و مدیریت حجم بالای داده برآید.



شکل۱-۱. پیچیدگی در حال افزایش سیستم

اکنون گلوگاه اصلی در طراحی با هماهنگی میان سرعت/عملکرد، کنترل تأخیر گیت نیست، بلکه تأخیرهای اتصال، مصرف توان و سرعت پایین گذرگاه سیستم گلوگاه‌های اصلی طراحی هستند. یک راه برای غلبه بر این گلوگاه‌ها، قرار دادن هسته‌های VLSI گوناگون، حافظه و پردازنده‌ها درون یک تراشه است. این کار زمان عکس‌العمل و تأخیرهای دسترسی به اطلاعات خارج تراشه را حذف می‌کند و در نتیجه باعث افزایش کارایی می‌شود.

در این راستا برای غلبه بر چالش‌های مذکور، مهندسان استفاده از یک الگوی طراحی جدید به نام سیستم بر تراشه "SoC" را آغاز کرده‌اند، شکل۱-۲. در الگوی طراحی SoC، تمام عملکردهای یک سیستم کامل درون یک تراشه سیلیکونی یکتا قرار داده می‌شود. یک چیپ SoC معمولی، ممکن است از ریزپردازنده‌ها که حاوی نرم‌افزار حاکم بر تراشه هستند، حافظه، مدارهای جانبی، قطعات جانبی (glue logic) و ماژول‌های آنالوگ تشکیل شده باشد (شکل۱-۳). الگوی طراحی SoC استفاده مجدد از هسته‌های IP سیلیکونی را ممکن می‌سازد. اکنون طراحان می‌توانند سیستم‌های کامل را با قرار دادن هسته‌های IP گوناگون به همراه یکدیگر درون یک تراشه یکتا بسازند. این کار منجر به کاهش زمان طراحی و هزینه‌های ساخت می‌شود. مجتمع‌سازی کامل تمام عملکردها درون یک تراشه به معنای کارایی و سرعت بهتر، توان کمتر و قابلیت اطمینان بیشتر است [۱۰]. به طور خلاصه، SoC تنها مجتمع نیست. تمام طراحی معماری به محدوده‌ی وسیعی اطلاعات و مهارت از سیستم نرم افزار، سخت افزار، منطق، مود – مختلط، فرکانس رادیویی، EDA، بسته‌بندی، آزمایش و ساخت تولید نیاز دارد.



شکل۱-۲. نمایی از پیچیدگی یک SoC

در مراحل اولیه طراحی SoCها، نه تنها سخت‌افزار، بلکه کل سیستم شامل نرم‌افزار نیز باید مدل شوند تا طراحی تأیید و معتبر شود.

همان‌طور که گفته شد، الگوی طراحی SoC با پیشرفت‌های جدید در تکنولوژی ساخت IC ممکن شده‌است، بنابراین با توانایی قرار دادن ترانزیستورهای بیشتر درون یک تراشه، می‌توان عملکردهای بیشتری را نیز درون یک چیپ یکتا قرار داد.

در طراحی‌های سیستم بر تراشه "SoC" با پیچیدگی بالا، تکنیک‌های ارزیابی IC در روند طراحی نقش مهمی را ایفا می‌کنند. یک بخش قابل ملاحظه از بررسی SoC پیچیده باید در سطح سیستم انجام شود. این مسأله یک چالش منحصر به فرد را برای نرم‌افزارهای منطقی مطرح می‌کند، چرا که اتصالات و سیم‌های داخلی طرح، در سطوح پایین به آسانی قابل دسترسی نیستند.

با افزایش روزافزون پیچیدگی SoC، روش‌های قدیمی ارزیابی و عیب‌یابی طرح برای تأیید عملکرد صحیح سیستم، که به صورت جداگانه با استفاده از شبیه‌سازهای سخت‌افزار HDL برای بخش سخت‌افزار دیجیتال، شبیه‌سازهای آنالوگ و سپس پیاده‌سازی نرم‌افزار، انجام می‌شدند، کارایی بسیار پائینی دارند. زمان‌های شبیه‌سازی طولانی و گاهی غیرعملی امکان استفاده از تست واقعی در صنعت الکترونیک را با این روش‌ها، بسیار محدود می‌کند.

شکل۱-۳. نمایی از پیچیدگی یک SoC