ارائه محیطی مبتنی بر TLM-2.0 برای مدل‌سازی کارآیی معماری ارتباطات در سیستم‌های برچیپ

مینا زلفی لیقوان[۱]،مربی و دانشجوی دکتری، ضیاءالدین دایی کوزه‌کنانی[۲]، دانشیار

۱-دانشکده مهندسی برق و کامپیوتر- دانشگاه تبریز- تبریز-ایران-mzolfy@tabrizu.ac.ir

۲-دانشکده مهندسی برق و کامپیوتر- دانشگاه تبریز- تبریز-ایران-zdaie@tabrizu.ac.ir

چکیده : یکی از مهم‌ترین موضوعات مطرح در زمینه طراحی سیستم‌های برچیپ، طراحی معماری ارتباطی[۱] در آن‌ها می‌باشد. پس از انتخاب عناصر محاسباتی مورد نیاز برای یک سیستم برچیپ، تصمیم‌گیری در مورد نحوه برقراری ارتباط بین آن عناصر، چه از لحاظ توپولوژی ارتباطی و چه از لحاظ پروتکل‌های ارتباطی یکی از مراحل مهم و موثر در طراحی کل سیستم می‌باشد. لذا در اغلب پروسه‌های طراحی لازم است که در مرحله مذکور، با هدف رسیدن به یک انتخاب مناسب کاوش فضای حالت[۲] و مقایسه حالات مختلف صورت گیرد.

در این مقاله محیطی مبتنی بر OSCI TLM-2.0 ارائه شده است که امکان ایجاد خودکار و دینامیک یک معماری ارتباطی را به صورت مجرد فراهم می‌کند. در این محیط بدون اینکه نیازی به مدل‌سازی جداگانه حالات مختلف باشد، مشخصات معماری مورد مطالعه در قالب فایل‌هایی با فرمت مشخص گرفته شده و به صورت خودکار مدل مجرد بر مبنای فایل‌های دریافتی ایجاد شده، شبیه سازی می‌شود.

واژه های کلیدی: سیستم برچیپ، معماری ارتباطی برچیپ[۳]، مدل‌سازی در سطح تراکنش[۴]، مدل‌سازی ارتباطات

TLM-2.0 Based Environment for Performance Modeling of on Chip Communication Architecture

Mina- Zolfi Lighvan[1] , Ziyaaldin- Daei Koozehkonani[2]

1, 2 -Faculty of Electrical and Computer Engineering- University of Tabriz

Abstract: In this paper an environment implemented for performance modeling of on chip communication architecture is presented. Our goal in this development is providing an environment for fast on chip communication architecture modeling and simulation. In this environment the computation details of the components are ignored and they are modeled with a general timing parameter in a higher level of abstraction. The implemented environment generates an abstract dynamic model of communication architecture in OSCI TLM-2.0 and simulates it using SystemC simulation kernel.

The most important advantage of this environment is its speed in simulation of communication architectures. On one hand, the computation details are ignored and on the other hand, the model is built and simulated in C++ without any other software interfaces.

Keywords: System on Chip, On Chip Communication Architecture, Transaction Level Modeling, Communication Modeling

تاریخ ارسال مقاله: ۹۰/۸/۳۰

تاریخ اصلاح مقاله: ۹۱/۱/۲۶

تاریخ پذیرش مقاله: ۹۱/۱/۳۰

نام نویسنده مسئول: مینا زلفی لیقوان

نشانی نویسنده مسئول: ایران- تبریز- بلوار ۲۹ بهمن- دانشگاه تبریز- دانشکده مهندسی برق و کامپیوتر

۱- مقدمه

رشد روزافزون تکنولوژی ساخت ادوات الکترونیکی و به تبع آن تقاضاها و نیازمندی‌های ایجاد شده برای سیستم‌های دیجیتالی، باعث بروز چالش‌های جدید در این حوزه از علم و فن آوری شده است. امروزه حضور سیستم‌های برچیپ[1] در اغلب سیستم‌های الکترونیکی مشاهده می‌شود و همین امر نشان دهنده اهمیت آن‌ها و عاملی برای وسعت و گسترش پژوهش‌ها و تحقیقات انجام گرفته در این زمینه می‌باشد. با وجود خودکار سازی قسمت‌های عمده ای از پروسه طراحی سیستم‌های دیجیتال ، کاوش فضای حالت، در اکثر مراحل طراحی، امری اجتناب ناپذیر می‌باشد. لذا لازم می‌گردد در مراحل مذکور، کاوش فضای حالت تحت نظارت طراح و یا توسط کامپیوتر انجام گیرد.

در حیطه طراحی ارتباطات یک سیستم برچیپ، به منظور کاوش در بین حالات موجود، معمولاً برای هر حالت مورد نظر، عناصر محاسباتی و ارتباطی، تحت معماری ارتباطات انتخاب شده، شبیه سازی می‌شوند. این فرایند به دلیل حجم بالای عناصر و پیچیدگی سیستم، زمان‌بر بوده و در کل باعث کند شدن روند طراحی می‌گردد. به منظور مقابله با این ضعف و با هدف مدل‌سازی کارایی[2] در این کار پژوهشی محیطی ارائه شده است که امکان پیکربندی و شبیه سازی مدلی مجرد از معماری ارتباطات را فراهم می‌سازد. در این ساختار جزئیات عناصر محاسباتی مطرح نشده و تنها به صورت مجرد، برخی ویژگی‌های زمان‌بندیِ آن‌ها، در شبیه سازی منظور می‌گردد. بدین ترتیب زمان شبیه سازیِ عناصر محاسباتی حذف شده و روند شبیه سازی حالات مختلف و مقایسه‌ی آن‌ها بسیار سریع‌تر انجام می‌گیرد.

در این محیطِ خودکار، اطلاعاتِ مربوط به اتصالات سیستم و همچنین اطلاعات کلی در مورد تأخیر عناصر در قالب تعدادی فایل متنی داده می‌شود. نرم افزارِ ارائه شده که در زبان C++ و با بهره‌مندی از امکانات SystemC و ِ OSCI TLM-2.0 پیاده سازی شده است، شبکه ارتباطی بین عناصر را به طور خودکار و در TLM-2.0 پیکربندی می‌کند. سپس بر اساس ترافیک ارتباطی که باز به صورت یک فایل متنی بیان می‌گردد، کل معماری ارتباطات را شبیه سازی کرده و نتیجه‌ی شبیه‌سازی را با زمان‌بندی در فایل خروجی می‌دهد.

در ادامه مقاله، ابتدا در بخش دوم توضیحاتی در مورد مدل‌سازی در سطح تراکنش و استاندارد OSCI TLM-2.0 خواهد آمد. سپس در بخش سوم سیستم‌های برچیپ و معماری ارتباطی درون آن‌ها توضیح داده خواهد شد. در بخش چهارم مروری بر کارهای گذشته خواهد آمد. بخش پنجم مقاله به ارائه محیط پیاده سازی شده می پردازد. در بخش ششم نمونه ای از نتایج آزمایشات انجام شده روی محیط مذکور تشریح خواهد شد و در نهایت با قسمت نتیجه گیری این مقاله خاتمه خواهد یافت.

۲- مدل‌سازی در سطح تراکنش

مدل‌سازی کارآمد ارتباطات، یکی از حساس‌ترین مراحل در طراحی سیستم‌های برچیپ و کاوش فضای حالت مربوطه می‌باشد. در واقع هدف از این مدل‌سازی، رسیدن به یک سیستم ارتباطی سریع و بهینه می‌باشد. اخیراً مدل‌سازی در سطح تراکنش به دلیل سرعت بالایی که در شبیه سازی ارتباطات دارد به کرات در این زمینه مورد استفاده قرار گرفته است. در حال حاضر یکی از استانداردهای ارائه شده برای مدل‌سازی در سطح تراکنش، استاندارد OSCI TLM-2.0 می‌باشد که در ماه ژوئن سال ۲۰۰۸ ارائه شد[۲]. TLM-2.0 در واقع یک کتابخانه به زبان C++ بوده و شامل پیاده سازی کلاس‌هایی است که این کلاس‌ها روی کتابخانه SystemC[۳، ۴] قرار می‌گیرد. SystemC نیز کتابخانه ای در زبان C++ می‌باشد که امکان شبیه سازی هم‌روند و زمان‌بندی شده را برای مدل‌های سخت افزاری فراهم می‌سازد.

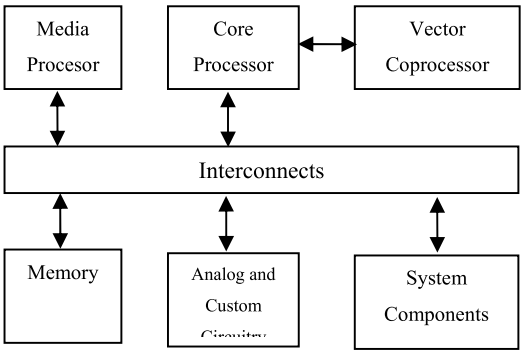
مدل‌سازی در سطح تراکنش زمانی می‌تواند نقاط قوت خود را نشان دهد که در سیستم مورد نظر جزییات ارتباطی و محاسباتی از هم جدا شده باشند. در این حالت تراکنش‌هایی که در درون معماری ارتباطات در سیستم مبادله می‌شوند توسط TLM مدل می‌شوند. در استاندارد TLM-2.0 ارتباطات بین ماجول‌ها و در واقع همان تبادل تراکنش‌ها از طریق تعدادی فراخوانی تابع که واسط حامل نامیده می‌شوند، عملی می‌گردد.

۳- سیستم‌های برچیپ

یک سیستم برچیپ در واقع مجموعه ای از IP Core[3] ها، پردازنده‌ها، حافظه‌ها و اتصالات بین آن‌ها را در برمی گیرد که برای پاسخ دهی به

نیاز های گروهی از کاربردها به روی یک تراشه واحد قرار گرفته‌اند[۱]. شکل ۱ برخی عناصر پایه یک سیستم برچیپ را نشان می‌دهد. عناصر مذکور تشکیل شده‌اند از تعدادی پردازنده نامتجانس[۴] که توسط یک سیستم ارتباطی به یک یا چند حافظه متصل شده‌اند. از آنجا که در سیستم‌های برچیپ، کلیه فرایندهای مربوط به یک کاربرد روی چیپ واحد صورت می‌گیرد، این سیستم‌ها معمولاً دارای هسته‌های آنالوگ نیز هستند-مثلاً برای تبدیل برخی سیگنال‌های آنالوگ به دیجیتال و برعکس.

با بالا رفتن پیچیدگی سیستم‌های برچیپ و افزوده شدن بر تعداد هسته‌های روی آن، تأثیر نقش معماری ارتباطات روی چیپ و نحوه اتصال هسته‌ها به یکدیگر در کارایی کل مدار، مهم‌تر می‌گردد[۵]. همان قدر که انتخاب معماری ارتباطات مناسب می‌تواند کارایی سیستم را بالا ببرد، انتخاب نادرست آن نیز می‌تواند نقش مخربی در عملکرد کلی سیستم داشته باشد. در طراحی معماری ارتباطات دو مسئله مطرح می‌گردد: انتخاب توپولوژی مناسب برای برقراری اتصالات و انتخاب پروتکل مناسب برای برقراری و مدیریت ارتباطات مورد نیاز، که هر دو مورد نقش مهمی در طراحی نهایی دارند و در انتخاب مناسب آن‌ها باید نهایت دقت به عمل آید.



شکل ۱ :مدل پایه یک سیستم برچیپ[۱]

۴- کارهای گذشته

با توجه به پیچیدگیِ بالای سیستم‌های دیجیتالی، شبیه سازیِ آن‌ها با دقت بالا و در سطوح پایین بسیار زمان‌بر بوده و عملاً غیر ممکن می‌باشد. به همین دلیل برای بررسی و آنالیزِ این سیستم‌ها ، کاوش فضای حالت بر مبنای مدل‌های سطح بالا مورد توجه قرار می-گیرد. بدین ترتیب وسعت فضای حالت پیاده‌سازی‌های ممکن در همان مراحلِ اولیه‌ی روند طراحی کاهش یافته، هزینه و زمان طراحی و ساخت کمتر می‌شود.

تا کنون برای نمایش SoCها و مدل‌سازیِ آن‌ها، از ابزارها و تکنیک‌های مختلفی استفاده شده است ولی در پژوهش‌های انجام شده توجه چندانی به مدل‌سازی ارتباطات نشده است. لذا در اغلب کارهای انجام شده، تکیه بر مدل‌سازی قسمت‌های محاسباتیSoC ها، بالاخص MPSoC[۵]ها بوده است. پیش بینی مجمع ITRS[۶] در مورد روند افزایش تعداد پردازنده‌ها در SoCها بیانگر چهار برابر شدن تعداد پردازنده‌ها در هر گره تکنولوژی می‌باشد[۶] و این موضوع نشانگر اهمیت معماری ارتباطی در MPSoCها می‌باشد. در این قسمت برخی از تکنیک‌های مذکور که در پروژه های مختلف بکار گرفته شده‌اند مورد بررسی قرار گرفته است.

در ساله‌ای گذشته، پروژه های متعددی در زمینه‌ی مدل‌سازی سیستم‌های برچیپ توسط UML [7] انجام شده است و بررسی نتایجِ حاصل از این پروژه‌ها نشان می‌دهد که UML به تنهایی پاسخگو نیست و به منظور بهره‌برداری صحیح و مناسب از UML برای مدل‌سازیِ سیستم‌ها، لازم است تغییراتی در آن اعمال شده و ویژگی‌هایی به آن افزوده گردد. در [۸] استفاده توأم از دو ابزار مهمِ Actor oriented modeling [9] و UML [10] برای توصیف و مدل‌سازی در سطح سیستم توصیه شده است. UML متشکل از خانواده‌ای از نمادهای تصویری می‌باشد که از تعدادی از زبان‌های مدل‌سازی مشتق شده است و به منظور استفاده در حیطهِ مهندسی نرم افزار ارائه گردیده است.

در مرجع [۱۱]، SDG[۷] به عنوان یک ابزار مناسب برای مدل‌سازی واسط HW/SW و همچنین به عنوان ابزاری برای تولید SoC مربوطه ارائه شده است. مدل SDG بر مبنای سرویس‌ها پایه گذاری شده است. یک واسط توسط مجموعه سرویس‌هایی که نیاز دارد و یا تأمین می‌کند، معرفی می‌گردد. ایده SDG از روش پیشنهادیِ Zitterbartبرای ترکیب واسط‌ها بر اساس سرویس، اتخاذ شده است. در[۱۲] نیز روش مذکور برای نرم افزار های فراگیر در سیستم‌های ارتباطات مخابراتی استفاده شده است. یک مدل واسطِ مبتنی بر سرویس از دو مجموعه‌ی گذرگاه منطقی که توسط یک SDG به هم وصل شده‌اند، تشکیل می‌شود.

در مرجع [۱۳]، ساختمان داده SIR[۸] برای مدل‌سازی در سطح سیستم ارائه شده است. SIR مستقل از گرامر زبان‌های توصیف در سطح سیستم بوده و قابلیت بیان سیستم در سطوح مختلف تجرید را دارد. ساختمان داده SIR شامل تعدادی کلاس در C++ می‌باشد.

مدل OSM[۹] به منظور توصیف پردازنده‌ها ارائه شده است[۱۴، ۱۵]. در این مدل توصیف پردازنده‌ها در دو لایه‌ی عملیاتی و سخت افزاری بیان می‌شود.در لایه سخت افزاری هر یک از عملیات ماشین توسط یک FSM[۱۰] مدل می‌شوند. FSMهای مذکور توسط پروتکل تراکنش توکن، با لایه سخت افزاری ارتباط برقرار می‌کنند. در لایه سخت افزاری، منابع داده و منابع ساختاری به شکل توکن‌ها مدل می‌شوند. در [۱۶] به منظور مدل‌سازی زیر سیستم‌های ارتباطی و کاوش در فضای حالت مربوطه از OSM استفاده شده است. در این روش مدل‌سازی ، امکان مدل‌سازی توأم قسمت‌های محاسباتی و ارتباطی در یک قالب واحد توسط MADL[۱۱] فراهم شده است[۱۷]. در روش ارائه شده برای مدل‌سازی از دو نوع عنصر: PE , OCA استفاده شده است. در این پروژه تکیه بیشتر روی قسمت محاسباتی می‌باشد و به همین دلیل روی مدل‌سازی قسمت ارتباطی کار زیادی انجام نشده است.

در مرجع [۱۸]، Colif به عنوان یک روش مدل‌سازی مبتنی بر شیء برای مدل کردن اتصالات داخلی نرم افزار و سخت افزار ارائه شده است. در مدل Colif از سه مفهوم پایه استفاده می‌شود: module, port و net . در این مدل عناصر نرم افزاری و سخت افزاری استفاده کننده و یا تأمین کننده سرویس‌های ارتباطی هستند(عناصر می‌توانند هم سرویس گیرنده باشند و هم سرویس دهنده). در روند مدل‌سازی Colif ابتدا یک معماری مجازی از سیستم ارائه می‌شود و سپس طی چند مرحله پالایش ارتباطات نرم افزار و سخت افزار، مدل قابل اجرای نهایی بدست می‌آید.

در مرجع [۱۹]، OSSS 2.0 به عنوان یک کتابخانه قابل سنتز برای سطح سیستم ارائه شده است. OSSS در واقع یک متدولوژی طراحی بر مبنای SystemC می‌باشد که مدل‌سازی شیء گرا را برای سیستم‌های سخت افزاری/نرم افزاری قابل سنتز فراهم می‌سازد. در OSSS زیر مجموعه قابل سنتز با افزودن المان‌هایی برای مدل‌سازی سطح بالا مانند متغیرهای مشترک، پلی مورفیسمو مدل‌سازی در سطح تراکنش توسعه داده شده است و در ضمن کتابخانه شبیه سازی و ابزار سنتز به نام Fossy[۱۲][۲۰] نیز در آن قرار داده شده است.

OSSS+R[۱۳] [۲۱] که در اصل شکل توسعه یافته OSSS می‌باشد، با افزودن المان‌ها زبان برای عناصر با قابلیت پیکربندی مجدد و نوبت دهی عناصر به وجود آمده است. هدف از ارائه OSSS+R ایجاد امکاناتی برای شبیه سازی و سنتز سیستم‌های با قابلیت پیکربندی مجدد در یک بستر شیء گرا می‌باشد[۲۲]. در OSSS+R با بهره‌مندی از قابلیت پلی مورفیسم زبان‌های شیء گرا امکان پیکربندی مجدد در حین شبیه سازی فراهم شده است.

CSP[۱۴] یک زبان رسمی برای توصیف الگوهای تعاملی در سیستم‌های هم‌روند می‌باشد[۲۳]. CSP برای اولین با در سال ۱۹۷۸ در مقاله ای توسط C. A. R. Hoare ارائه شد[۲۴]. توسط CSP سیستم‌ها به صورت فرآیند های مستقل از هم توصیف می‌شود که تعامل فرآیندها تنها با همگام سازی از طریق رخدادهای ارتباطیِ حامل پیغام می‌باشد. در CSP با استفاده از اعمال عملگرهای جبری روی فرآیندهای ساده، فرآیندهای پیچیده ساخته می‌شوند. CSP یک زبان برنامه نویسی نیست و در اصل برای بیان ارتباط فرآیندها ارائه شده است و قابلیت توصیف رفتار داخلی فرآیندها را ندارد. کتابخانه های متعددی برای زبان‌های متفاوت ایجاد شده است که قابلیت پیاده سازی عناصر CSP را فراهم می‌کنند[۲۵-۲۸]. در این کتابخانه‌ها از امکانات زبان میزبان در پیاده سازی هم‌روندی برای تعریف فرآیندهای CSP استفاده شده است.

در یک مدل محاسباتی KPN[29, 30][۱۵] ، شبکه ای از فرایند های هم‌روند وجود دارند که به شکل نقطه به نقطه و به واسطه FIFO های نامحدود با هم ارتباط برقرار می‌کنند. در چنین مدلی همزمانی بین فرآیندها توسط Blocking readها عملی می‌شود و فرآیندها به صورت مستقل از هم اجرا می‌شوند. در این مدل با توجه به اینکه کنترل به صورت توزیع شده انجام می‌گیرد، هیچ زمان‌بند عمومی وجود ندارد. از طرفی چون تبادل اطلاعات روی FIFOها توزیع شده است، حافظه عمومی وجود ندارد(حافظه ای وجود ندارد که توسط چند فرآیند قابل دسترسی باشد).در مرجع [۳۱]، از مدل KPN برای تصویر کردن یک کاربرد به روی سیستمی متشکل از یک میکروپروسسور و یک FPGA استفاده شده است..در این پژوهش مدل KPN به صورت اتوماتیک از کاربردی که به زبان Matlab نوشته شده است استخراج

شده و سپس مدل مذکور به طور سیستماتیک به روی سخت افزار هدف، تصویر می‌شود.

۵- محیط ارائه شده برای مدل‌سازی و شبیه سازی معماری ارتباطات

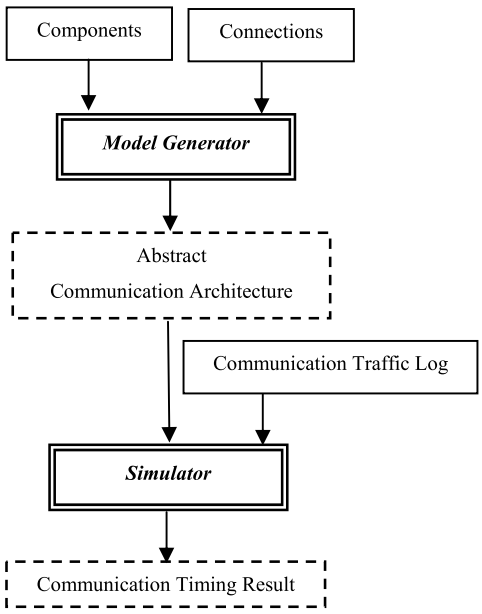
در این پژوهش، به منظور فراهم ساختن امکانات مدل‌سازی و شبیه سازی کارایی معماری ارتباطات در سیستم‌های برچیپ و کاوش فضای حالت مربوطه محیطی مبتنی بر TLM-2.0 پیاده سازی شده است. این محیط که به زبان C++ نوشته شده است، اطلاعاتی در مورد ساختار سیستم و ترافیک ارتباطات را می‌گیرد و به صورت اتوماتیک مدل دینامیکی از معماری ارتباطی را بدون اینکه نیاز به جزییات محاسباتی سیستم داشته باشد، ایجاد کرده شبیه سازی می‌کند.

این محیط قابلیت مدل‌سازی معماری‌های ارتباطات شامل اتصالات مستقیم (نقطه به نقطه[۱۶])، یک یا چند باس مشترک[۱۷] و همچنین پل‌های بین باس‌ها را دارد. در اثنای پیکربندی مدل مربوط به یک معماری ، به ازای هر یک از عناصر (باس‌ها، پل‌ها و عناصر محاسباتی موجود در ساختار)، نمونه‌هایی از کلاس‌های پیاده سازی شده گرفته می‌شود و اتصالات مورد نیاز از طریق سوکت‌های تعبیه شده در کلاس‌ها و بر اساس استاندارد TLM-2.0 بر قرار می‌گیرد. واسط حامل TLM-2.0 [۱۸] تراکنش‌ها را بین آغازگرها[۱۹] و هدف‌ها[۲۰] منتقل می‌کند. آغازگر ماجولی است که یک تراکنش را تولید کرده و آن را با فراخوانی یکی از متد های واسط ارسال می‌کند. متقابلاً هدف ماجولی است که به عنوان مقصد نهایی تراکنش نقش ایفا می‌کند. هر ماجول آغازگر باید دارای سوکت آغازگر باشد و هر ماجول هدف باید سوکت هدف داشته باشد که موارد مذکور در TLM-2.0تعریف شده‌اند. در پیاده سازی که در این مقاله ارائه شده از واسط‌های حامل non-blocking برای تبادل تراکنش‌ها استفاده شده است.

۵-۱- روند کار

شکل ۲ مراحل مختلفِ مربوط به نرم افزارِ پیاده شده را نشان می‌دهد. در این شکل جعبه های "Components"، "Connections" و "Communication Traffic Log" نشان دهنده‌ی ورودی‌های نرم‌افزار می‌باشند که هر کدام به صورت یک فایل متنی داده می‌شوند.

روند طراحی در محیط ارائه شده، دو مرحله عمده‌ی *ایجاد مدل* و *شبیه سازی*، را شامل می‌شود. حاصل مرحله‌ی اول یا مرحله‌ی *" ایجاد مدل (Model Generator)"*، یک مدل مجرد در TLM-2.0 می‌باشد که با عنوان مدل مجرد معماری ارتباطات یا "Abstract Communication Architecture Model" در شکل آمده و بر اساس دو ورودیِ "Components" و "Connections" بدست می‌آید. در مرحله‌ی دوم از این روند یا همان مرحله *" شبیه سازی (Simulation)"* مدلی که پیکربندی آن در مرحله قبل صورت گرفته، به همراه فایل ورودیِ "Communication Traffic Log" گرفته شده، شبیه سازی انجام می‌گیرد و نتیجه شبیه سازی در فایل خروجی ثبت می‌گردد.



شکل ۲ : مراحل اجرای نرم افزار پیاده سازی شده

۵-۱-۱-مرحله ایجاد مدل

در این قسمت همان طور که قبلاً نیز ذکر شد، عناصر موجود در سیستم و اتصالات بین آن‌ها از طریق فایل‌های متنی components.txt و connections.txt گرفته شده، بر مبنای اطلاعات مندرج در این فایل‌ها مدل مجرد از سیستم ارتباطی به صورت دینامیک ایجاد می‌گردد. ایجاد مدل ارتباطی خود در دو مرحله صورت می‌گیرد : ۱) مرحله نمونه‌گیری عناصر[۲۱] و ۲) مرحله تنظیم

اتصالات[۲۲]. در مرحله نمونه‌گیری عناصر، عنصرهای مورد نیاز بر اساس محتویات فایل components.txt و با استفاده از امکانات SystemC نمونه گیری می‌شوند. نمونه های مذکور فاقد جزئیات محاسباتی هستند و تنها دارای زمان‌بندی محاسباتی و آن هم به صورت کلی می‌باشند. در مرحله تنظیم اتصالات، اتصالات بین عناصر به شکلی که در فایل connections.txt آمده برقرار می‌گردد. در برقراری اتصالات بین عناصر از Socketها و nonblocking transport interface های ارائه شده در استاندارد TLM-2.0 استفاده شده است.

۵-۱-۲-شبیه سازی

در این مرحله، مدلی که در مرحله قبل ایجاد شده است، با استفاده از فایل ورودی comtraffic.txt و توسط هسته شبیه ساز SystemC، شبیه سازی می‌شود. به منظور شبیه سازی، فایلِ comtraffic.txt توسط محیط پیاده‌سازی شده، سطر به سطر خوانده می‌شود و طبق زمان‌بندی ذکر شده در هر سطر، به مدل اعمال می‌گردد. خروجی این مرحله یک فایل متنی می‌باشد که در آن زمان درخواستِ هر تبادل و زمان برقراری آن (که بستگی به ساختار سیستم و ترافیک مربوطه دارد) درج گردیده است.

۵-۲- پیاده سازی محیط

در ادامه توضیحاتی در رابطه با ماجول های عمده ای که برای پیاده سازی محیط مورد بحث ایجاد شده‌اند آمده است.

۵-۲-۱- ماجول CommunicationArchitecture

این ماجول در واقع مدل کل سیستم را شامل می‌شود، به طوری که در هر آزمایش دقیقاً یک نمونه از آن گرفته می‌شود. شکل ۳ کد سازنده ماجول CommunicationArchitecture و در اصل مراحل اولیه شروع به کار آن را نشان می‌دهد.

```
SC_CTOR(CommunicationArchitecture)
                                    {
        Resultfp.open("Result.txt");
              set_componentsbuses();
                   set_connection();
    SC_THREAD(communication_request);
                                    };
```

شکل ۳: کد سازنده ماجول CommunicationArchitecture

همان‌طور که می‌بینیم، پس از معرفی فایل خروجی، ابتدا عناصر مدل شامل عناصر محاسباتی و باس‌ها ایجاد می‌شوند. در مرحله بعد ارتباطات بین عناصر موجود برقرار می‌شود. سپس فرایند هم‌روند communication_request که وظیفه اعمال درخواست‌های ارتباطی را دارد، شروع به کار می‌کند. این فرایند درخواست‌های ارتباط بین عناصر را از فایل مربوط به ترافیک ارتباطات می‌خواند و طبق زمان‌بندی مشخص شده در فایل CommunicationTrafficLog، به سیستم اعمال می‌کند.

۵-۲-۲- ماجول component

در مدل‌های مجردی که توسط محیط پیاده سازی شده ایجاد می‌شوند، به ازای هر عنصر محاسباتی، نمونه ای از ماجول component گرفته می‌شود. این ماجول دارای سوکت‌های آغازگر و هدف می‌باشد که به واسطه آن‌ها می‌تواند با سایر عناصر سیستم، ارتباط بر قرار کند. تعداد سوکت‌های مورد نیاز هر نمونه و نحوه اتصال آن‌ها به صورت دینامیکی به واسطه‌ی متدهای set_componentsbuses()وset_connection() که به صورت توابع عضو ماجول CommunicationArchitecture پیاده شده و در سازنده آن فراخوانی می‌شوند مشخص می‌گردند. هر ماجول component یک فرایند هم‌روند به نام sense دارد. این فرایند حساس به رخدادهای ارتباطی ماجول بوده و بر اساس پارامترهای زمان‌بندی ماجول به این رخدادها پاسخ می‌دهد. لازم به ذکر است که مسئولیت فعال سازی رخدادهای ارتباطی به عهده ماجول CommunicationArchitecture می‌باشد.

در محیط پیاده‌سازی شده، هر پل ارتباطی موجود در معماری ارتباطات که به منظور برقراری ارتباط بین دو باس مشترک استفاده می‌شود، به عنوان یک عنصر محاسباتی با امکانات خاص در نظر گرفته

می‌شود و در واقع به ازای هر پل یک نمونه از ماجول component گرفته می‌شود.

۵-۲-۳- ماجول sharedbus

در مدل‌های ارتباطی، به ازای هر باس مشترک نمونه‌ای از ماجول sharedbus گرفته می‌شود. ماجول‌های sharedbus دارای سوکت‌های آغازگر و هدف می‌باشند که به منظور برقراری اتصالات مورد استفاده قرار می‌گیرند و تعداد و نحوه اتصال آن‌ها را ماجول CommunicationArchitecture مشخص می‌کند. هر نمونه از ماجول sharedbus دارای یک فرایند هم‌روند به نام arbiter می‌باشد که مسئولیت نوبت دهی در باس‌های مشترک و پاسخ دهی به درخواست‌های ارتباطی عناصر متصل به آن را دارد.

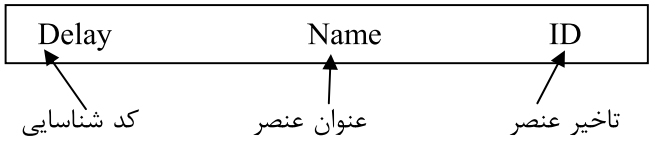
۵-۳- فایل‌های ورودی

در ابزار مورد بحث، ورودی‌ها به صورت فایل‌های متنی و با فرمت های قراردادی داده می‌شوند که توضیحات مربوطه در ادامه آمده است.

۵-۳-۱- فایل Components.txt

در این فایل عناصر سیستم معرفی می‌شوند که می‌توانند از دو نوع محاسباتی و باس باشند. سطر اولِ این فایل حاوی دو عدد صحیح است که به ترتیب تعداد عناصر محاسباتی و باس‌ها را مشخص می‌کند.

در سطور بعدی مشخصات عناصر نوشته می‌شوند که هر سطر مشخصات یک عنصر را شامل می‌شود. لازم به ذکر است در سطرهایی که مشخصات عناصر مشخص شده، ابتدا عناصر محاسباتی و بعد باس‌ها آمده‌اند. شکل ۴ فرمت داده‌ها را در هر سطر از فایل components.txt نشان می‌دهد.

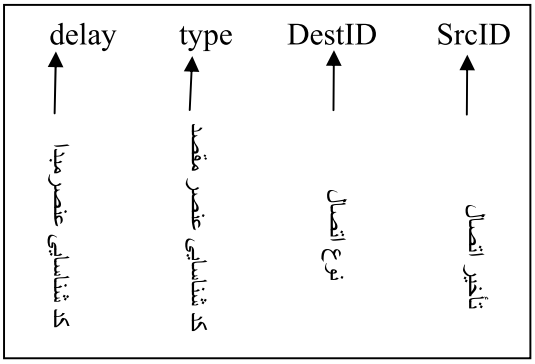


شکل ۴ : فرمت مورد استفاده برای معرفی یک عنصر در فایل Components.txt

۵-۳-۲- فایل Connections.txt

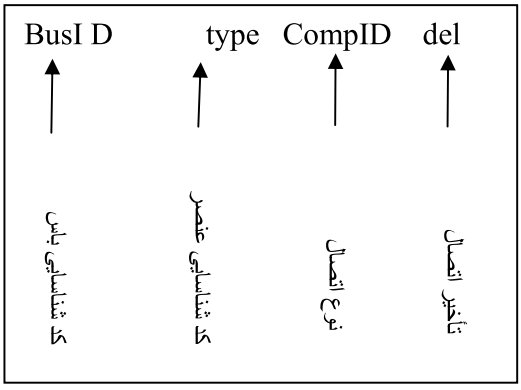
در این فایل اتصالاتِ واقع شده بین عناصر سیستم درج می‌شود. سطر اول از این فایل حاوی دو عدد صحیح است که اولی تعداد اتصالات مستقیم و دومی تعداد اتصالات غیر مستقیم را مشخص می‌کند. منظور از یک اتصال مستقیم، اتصالِ نقطه به نقطه[۲۳] بین دو عنصر محاسباتی و یا بین یک عنصر و یک پل می‌باشد و منظور از یک اتصال غیر مستقیم، اتصالِ یک عنصر محاسباتی یا یک پل با یک باس می‌باشد که می‌تواند منشأ ایجاد اتصال بین دو عنصر با واسطه یک باس باشد. در سطور بعدی از این فایل، ابتدا اتصالات مستقیم و سپس اتصالات غیر مستقیم قید شده است.

در شکل ۵ فرمت معرفی یک اتصال مستقیم در فایل Connections.txt آمده است. فیلد *"نوع اتصال"* در این فرمت می‌تواند ۰ و یا ۱ باشد. در صورتی که این فیلد حاوی مقدار ۰ باشد نشان دهنده اتصال یک طرفه از عنصر مبدأ به عنصر مقصد می‌باشد و در صورتی که مقدار ۱ داشته باشد نشان دهنده اتصال دوطرفه بین مبدأ و مقصد است.



شکل ۵ : فرمت مورد استفاده در فایل Connections.txt برای اتصالات مستقیم

در شکل ۶ فرمت معرفی یک اتصال غیر مستقیم در فایل Connections.txt آمده است. فیلد "نوع اتصال"در این فرمت می‌تواند ۰، ۱ و یا ۲ باشد. در صورتی که این فیلد مقدار ۰ داشته باشد نشان دهنده اتصال یک طرفه از باس به عنصر ، در صورتی که مقدار ۱ داشته باشد نشان دهنده اتصال یک طرفه از عنصر به باس و در صورتی که مقدار ۲ داشته باشد نشان دهنده اتصال دوطرفه بین باس و عنصر است.



شکل ۶ : فرمت مورد استفاده در فایل Connections.txt برای اتصالات غیر مستقیم

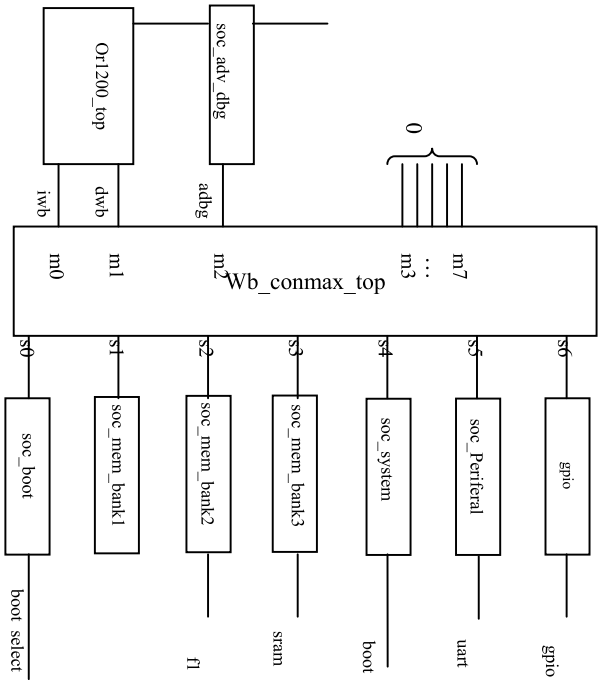
	0	SrcID	DestID	timestep
timestep	1	BusID	SrcID	DestID

شکل ۷ : فرمت مورد استفاده در فایل ComTraffic.txt

۵-۳-۳- فایل Comtraffic.txt

این فایل در واقع یک log file از تبادل اطلاعات و ارتباطات مورد نیاز در یک سیستم و در یک کاربرد خاص می‌باشد. در این فایل نیازهای ارتباطیِ بین عناصر به صورت زمان‌بندی شده قید می‌گردد. در سطر اول این فایل یک عدد صحیح نوشته می‌شود که تعداد ارتباطاتِ برقرار شده در داخل فایل را مشخص می‌کند. در سطور بعدی، هر سطر یک در خواست ارتباطی را معرفی می‌کند. هر درخواست ارتباطی می‌تواند از یکی از دو نوع ممکن باشد. اولین مقدار در هر سطر، نوع ارتباط را مشخص می‌کند. این مقدار اگر ۰ باشد، یعنی سطر مربوطه‌اش ارتباط از طریقِ یک اتصال مستقیم را مشخص می‌کند و اگر ۱ باشد یعنی ارتباط از طریق یک اتصال غیر مستقیم درخواست می‌گردد.

شکل ۷ فرمت سطرهای فایلِ comtraffic را برای هر دو نوع ارتباط نشان می‌دهد. فیلد timestep فاصله زمانی با سطر قبلی در فایل را نشان می‌دهد.



شکل(۸) : مدار OR1200_SOC

۶- مدار نمونه

در این قسمت عملکرد نرم افزار و فایل‌های ورودی و خروجیِ آن به همراه یک مثال بررسی خواهد شد. مثال مورد نظر OR1200 SOC می‌باشد که در واقع یک سیستم برچیپ نمونه، گرفته شده از مدل‌های OPEN Core می‌باشد. در شکل(۸ شماتیکی از اتصالات مدل اولیه این سیستم که در Open Core ارائه شده است نشان داده شده است. مدار مذکور شامل پردازنده OR1200 به عنوان پردازنده اصلی سیستم برچیپ، سه واحد حافظه و تعدادی عنصر دیگر (مطابق شکل ۸) برای راه اندازی سیستم و ارتباط با خارج از آن می‌باشد. همان‌طور که ملاحظه می‌شود، کلیه عناصر با تنها باس مشترک سیستم که از نوع wishbone conmax می‌باشد با سایر عناصر ارتباط پیدا می‌کنند و تنها یک ارتباط مستقیم در سیستم وجود دارد که بین پردازنده و واحد اشکال زدایی سیستم(که در شکل soc_adv_dbg نام‌گذاری شده است) برقرار شده است.

در شکل ۹ فایلِ معرفی عناصر سیستم، مربوط به آزمایش نمونه نشان داده شده است. در این نمونه ۹ عنصر محاسباتی و یک باس

وجود دارد که طبق فرمت و ترتیب قراردادی، اطلاعات مربوط به آن‌ها در فایل درج گردیده است.

```
9 1
0 OR1200  6
1 adv_dbg 5
2 sys_boot 4
3 mem_bank_1 10
4 mem_bank_2 15
5 mem_bank_3 20
6 system 2
7 peripheral 30
8 gpio 30
0 wishbone_conmax 1
```

شکل ۹ : محتویات فایل components.txt مربوط به Or1200_SOC

در شکل ۱۰ فایلِ معرفی اتصالات بین عناصر سیستم مربوط به مثال مورد بحث نشان داده شده است. در این نمونه یک ارتباط مستقیم بین عناصر OR1200 با کد شناسایی 0 و adv_dbg با کد شناسایی 1 وجود دارد و سطر دوم نشان دهنده‌ی این اتصال است. سایر اتصالات از طریق باس برقرار شده است که در سطرهای بعد طبق فرمت قراردادی مربوط در فایل درج گردیده است.

به منظور بررسی حالات مختلف از معماری ارتباطی داخل سیستم نمونه، علاوه بر شبیه سازیِ مدار اصلی که در شکل(۸ آمده است، دو حالت دیگر نیز برای ارتباطات داخل این مدار بررسی شد. جدول ۱ نتایج حاصله را نشان می‌دهد. حالت A نشان دهنده ِمدار اولیه می-باشد. در حالت B علاوه بر باس مشترکی که در سیستم وجود دارد، باس دیگری نیز افزوده شده است به طوری که ارتباط بین عناصر با کد ۴و۷ و همچنین ارتباط بین عناصر با کد ۵و۸ از طریق باس جدید تحقق می‌یابد. در حالت C باس جدید همان‌طور که در حالت B معرفی شد، حفظ گردیده با این تفاوت که این بار ارتباط بین عناصرِ ۷و ۵ و همچنین ارتباط بین عناصر ۳و۸ توسط باس جدید عملی می‌گردد.

با اعمال ورودی یکسان(CommTraffic.txt) حاوی ۳۰۰ درخواست ارتباطی به هر سه نمونه، دیده می‌شود که مدار C سریع‌تر از B و مدار B سریع‌تر از A می‌تواند درخواست‌های اعمال شده را انجام دهد.

جدول ۱ : زمان پاسخگویی به درخواست‌های ارتباطی یکسان در سه نمونه با معماری ارتباطی متفاوت از مدار OR1200_SOC

معماری مورد شبیه سازی	زمان مورد نیاز برای تکمیل شدن ارتباطات درخواست شده
OR1200_SOC_A	4913 ns
OR1200_SOC_B	4531 ns
OR1200_SOC_C	4309 ns

۷- نتیجه آزمایش‌ها

هدف از پیاده سازی ابزاری که توضیح داده شد ،فراهم ساختن زمینه ای برای مقایسه معماری‌های ارتباطی مختلف در یک SOC و کاوش فضای حالت مربوطه می‌باشد. از آنجا که هدف نهایی این پروژه، ارائه ساختاری برای نمایش و مدل کردن ارتباطات داخل یک SOC و همچنین سنتز آن با هدف بالا بردن سرعت می‌باشد، ابزار پیاده شده می‌تواند این مسیر را هموارتر نموده در مقایسه‌ی حالات مختلف مورد استفاده قرار گیرد.

ابزار ارائه شده در واقع کارایی یک معماری را در ازای ترافیک ارتباطی حاکم در آن می‌سنجد. به منظور بررسی توانمندی این ابزار در راستای هدفی که دنبال می‌کنند، آزمایش‌هایی بر روی آن و با استفاده از مجموعه محک[۲۴] ارائه شده توسط گروه تحقیقاتی دانشگاه صنعتی تمپر انجام شده است[۳۲، ۳۳]. مجموعه مذکور شامل سیستم‌های متفاوتی با مشخصات ارائه شده در جدول ۲ می‌باشد که کلیه این سیستم‌ها در XML و با مشخصات ارتباطی و ترافیک بین عناصر توصیف شده‌اند. طی آزمایشات انجام شده، معماری سیستم‌های مورد آزمایش از فرمت XML استخراج شده و در قالب فایل‌های ورودی محیط پیاده سازی شده در آمده‌اند. سپس با اعمال تغییراتی در ساختار معماری ارتباطی سیستم اولیه، شبیه سازی‌های مختلفی انجام شده است که جدول ۲ نتایج حاصله را نشان می‌دهد.

آزمایشات انجام شده و نتایج حاصل از آن‌ها نشان می‌دهد که ابزار ارائه شده قابلیت سنجش کارایی سیستم‌ها به ازای معماری‌های ارتباطی متفاوت را دارا بوده و می‌تواند در زمینه مذکور مورد استفاده قرار گیرد. لذا با توجه نتایج حاصل از شبیه‌سازی ساختارها و

معماری‌های مختلف ارتباطی و بر اساس شرایط و محدودیت‌های پیاده سازی می‌توان طراحی مناسب را انتخاب نمود.

```
1 9
0 1   1 0
0 0   2 0
0 1   2 0
0 2   2 0
0 3   2 0
0 4   2 0
0 5   2 0
0 6   2 0
0 7   2 0
0 8   2 0
```

شکل ۱۰ : محتویات فایل connections.txt مربوط به Or1200 SOC

۸- نتیجه گیری

در این مقاله محیطی برای مدل‌سازی و شبیه سازی کارایی معماری ارتباطی سیستم‌های برچیپ پیاده سازی شده است. ابتکاری که در این پژوهش صورت گرفته استفاده از امکانات C++, SystemC و TLM-2.0 در ایجاد مدل‌های دینامیکی با قابلیت شبیه‌سازی هم‌روند می‌باشد. در محیط مذکور اطلاعاتی در مورد ساختار ارتباطی سیستم و همچنین ترافیک ارتباطی پیش بینی شده برای یک کاربرد، در قالب فایل‌های متنی به ابزار داده می‌شود. این محیط مدل مجردی از معماری ارتباطی سیستم را به طور دینامیک در TLM-2.0 سنتز می‌کند و با استفاده از ترافیک داده شده آن را شبیه سازی می‌نماید.

از مزایای عمده این محیط سرعت بالای آن در شبیه سازی و به تبع آن امکان کاوش سریع‌تر فضای حالت می‌باشد. از آنجا که جزییات محاسباتی در مدل‌های ایجاد شده وارد نمی‌گردد و از طرفی مدل در سطح تراکنش در C++ و بدون اعمال واسطهای نرم‌افزاری شبیه سازی می‌شود، سرعت شبیه سازی به طور چشمگیر بهبود می‌یابد. یکی دیگر از مزایای این محیط، قابلیت بالای آن در پیاده سازی پروتکل‌های مختلف می‌باشد. در این محیط کلیه الگوریتم‌ها به زبان C++ پیاده سازی می‌شوند. رواج گسترده این زبان و همچنین قابلیت‌های شناخته شده کمک می‌کند تا محققین به سهولت بتوانند از ابزار ارائه شده برای بررسی و پیاده سازی الگوریتم‌ها، روش‌ها و پروتکل‌های ارتباطی مختلف استفاده کنند..

جدول ۲ : نتایج حاصله از آزمایشات انجام شده روی تعدادی از سیستم‌های مجموعه محک

Application	#task	#edges	Avg #dst	Max #dst	Experimental Results				
					#Communication Request	Different Structure			
						Point to Point	Shared Bus	Shared Bus + Point to Point	
UMTS Receiver[1]	7	11	1.6	6	800	1746	3700	(1B+3P)[5] 1960	
OFDM Receiver[2]	7	12	1.7	4	800	1092	6224		
Channel Equalizer[3]	12	20	1.7	4	1024	2610	5834	(1B+1P)5388	(2B+2P)4289
MWD[4]	12	22	2.4	4	1200	2058	5975	(1B+1P)4869	
VOPD[5]	12	14	1.2	2	1200	3375	6689		

1. UMTS for RAKE Receiver model[34]
2. OFDM (HiperLan/2) receiver model[35]
3. Channel-Equalizer[36]
4. Multiwindow Display, (MWD) model[37]
5. Video Object Plane Decoder, VOPD model[37]
6. One Shared Bus + One Point to Pont Connection

مراجع

[1] Flynn, Michael J. and Luk, Wayne, Computer System Design: System-on-Chip. 2011: wiley. 360.

[2] Aynsley, John, OSCI TLM-2.0 USER MANUAL. 2008: Open SystemC Initiative (OSCI.(

[3] Donavan, J, Bunton, B, and Keist, A, SystemC: From the Ground Up(Second Edition). 2009: Springer. 244.

[4] Grotker, T, Liao, S, Martin, G., and Swan, S, System Design with SystemC 2002: Springer. 240.

[5] Pasricha, Sudeep and Dutt, Nikil, On-Chip Communication Architectures (System on Chip Interconnect). 2008: Morgan Kaufmann Publisher.

[6] SOC System Driver Section, www.public.itrs.net.

[7] J. Rumbaugh, I. Jacobson and G. Booch, The Unified Modeling Language Refrence Manual. 1998: Addison-Wesley.

[8] Indrusiak, L. and Glesner, A. SoC Specification using UML and Actor-Oriented Modeling in Baltic Electronics Conference, 2006 International. 2006. Tallin.

[9] E. A. Lee , S. Neuendorffer and M. J. Wirthlin, Actor Oriented Design of Embedded Hardware and Software Systems. Journal of Circuits Systems and Computers, 2003. Vol. 12(No. 3): p. 231-260.

[10] Fowler, M., UML Distilled, 3rd edition. 2003: Addison-Wesley.

[11] A. Sarmento, L. Kriaa, A, Grasset, M. W. Youssef, A. Bouchhima, F. Rousseau, W. Cesario and A. A. Jerraya. Service Dependency Graph, an Efiicient Model for Hardware/Software Interface Modeling and Generation for SOC Design. in CODES+ ISSS 2005. 2005. Jersy City, New Jersy, USA.

[12] L. Kai, D. Gajski. Transaction Level Modeling: An Overview. in CODES+ISSS '03. 2003.

[13] Viskic, I and Domer, R. A Flexible, Syntax Independent Representation(SIR) for System Level Design Models. in 9th EUROMICRO conference on Digital System Design. 2006.

[14] Qin, Wei, Rajagopalan, S, and Malik, Sharad. A Formal Concurrency Model Based Architecture Description Language for Synthesis of Software Development Tools. in ACM SIGPLAN/SIGBED. 2004.

[15] Qin, Wei and Malik, Sharad. Flexible and Formal Modeling of Microprocessor with Application to Retargetable Simulation in Design Automation and Test in Europe. 2003.

[16] Zhu, Xinping, Qin, Wei, and Malik, Sharad, Modeling Operation and Microarchitecture Concurrency for Communication Architecture with Application to Retargetable Simulation. IEEE Transaction on Very Large Scale Integration (VLSI) Systems. 2006. Vol. 14(No. 7): p. 707-716.

[17] Qin, Wei, Modeling and Description of Embedded Processors for the Development of Software Tools 2004, Princeton, NJ: Princeton, University. p. 216

[18] Cesario, W., Nicolescu, G, Gautheir, L., Lyonnard, D., and Jerraya, A. Colif: a Multilayer Design Representation for Application Specific Multiprocessor System-on-Chip Design. in International Workshop on rapid System Prototyping. 2001. California, USA.

[19] Brunzema, C, Grabbe, C, Gruttner, K, Hartmann, P. A, Herrholz, A, Kleen, H, Oppenheimer, F, Schallenberg, A, Stehno, C, and Schubert, T, OSSS 2.0- A Library for Synthesisable System Level Model in SystemC. 2007.

[20] Fossy Synthesiser http://fossy.offis.de

[21] Schallenberg, A, Nebel, W, Herrholz, A, and Hartmann, P. A. OSSS+R: A framework for application level modelling and synthesis of reconfigurable systems. in IEEE Design automation and test in europe. 2009.

[22] Schallenberg, A, Nebel, W, and Oppenheimer, F. OSSS+R: Modelling and Simulating Self-Reconfigurable Systems. in FPL'06. 2006.

[23] Hoare, C. A. R, Communicating Sequential Processes. 2004: Prentice Hall International Series in Computing Science. 260.

[24] Hoare, C. A. R, Communicating Sequential Processes. ACM, 1978. Vol. 28(No. 8): p. 666-677.

[25] CSP Support for LISP http://www.cliki.net/csp.

[26] CSP for Java(JCSP) http://www.cs.kent.ac.uk/projects/ofa/jcsp/

[27] The Kent C++ CSP Library http://www.cs.kent.ac.uk/projects/ofa/c++csp.

[28] Fernando, H, Sanchez, P, and Villar, E. Modeling of CSP, KPN and SR Systems with SystemC .in FDL'03. 2003.

[29] Kahn, G. The Semantics of a Simple Language for Parallel Programming. in IFIP Congr. 1974.

[30] Lee, E. A and Parks, T. M, Dataflow process Networks. Proceedings of the IEEE, 1995. **83**(5): p. 773-799.

[31] Stefanov, T, Zissulescu, C ،Turjan, A, Kienhuis, B, and Deprettere, E. System Design Using Kahn Process Networks: The Compaan/Laura Approach. in Date'04. 2004.

[32] Salminen, Erno, Kangas, Tero, Lahtinen, Vesa, Riihimäki, Jouni, Kuusilinna, Kimmo, and Hämäläinen, Timo D., Benchmarking Mesh and Hierarchical Bus Networks in System-on-Chip Context. Journal of System Architectures, 2007.

[33] Kangas, Tero, Methods and Implementations for Automated System on Chip Architecture Exploration, in Tampere University of Technology. 2006.

[34] Wolkotte, P.,"Explorationwithingthe Network-On-ChipParadigm, in University of Twente.

[35] Rauwerda, R., Multi-standard adaptive wireless communication receivers, in University of Twente.

[36] Moonen, A. , Bekooij, M. , van den Berg, R. , and van Meerbergen, J. , Evaluation of the throughput computed with a dataflow model -A case study. 2007, ES Reports.

[37] Bertozzi, D, Jalabert, A., Murali, S., Tamhankar, R, Stergiou, S, Benini, L, and De Micheli, G, NoC synthesis flow for customized domain specific multiprocessor systems-on-chip. IEEE Trans. on Parallel and Distributed Systems, 2005. Vol. 16(No. 2): p. 113-129.

زیرنویس‌ها

[1] .SoC: System on Chip

[2] .Performance Modeling

[3] .IP Core: Intellectual Property Core

[4] Heterogeneous Processor

[5] MPSoC: Multi Processor System on Chip

[6]. ITRS: International Technology Roadmap for Semiconductors

[7] .SDG: Service Dependency Graph

[8] .SIR: Syntax Independent Representation

[9] .OSM: operation state machine

[10]. FSM: Finite State Machine

[11] .MADL: Mescal Architecture Description Language

[12]. FOSSY: Functional Oldenburg System Synthesiaer

[13] .OSSS+R: Oldenburg System Synthesis Subset + Reconfiguration

[14]. CSP: Communicating Sequential Processes

[15]. KPN: Kahn Process Network

[16]. Point to Point

[17]. Shared Bus

[18] .TLM-2.0 transport interface

[19]. Initiators

[20]. Targets

[21]. Component Instantiation

[22]. Connection Setting

[23]. Point to point

[24]. Benchmark