درستی‌سنجی صوری معماری مدیریت توان در سطح سیستم برای پردازنده‌های مدرن

سیدرضا شرفی نژاد[1] بیژن علیزاده[2]

۱- دانشجوی دکتری- پردیس دانشکده‌های فنی دانشگاه تهران - دانشکده مهندسی برق و کامپیوتر- تهران - ایران
r.sharafi@ut.ac.ir

۲- دانشیار- پردیس دانشکده‌های فنی دانشگاه تهران - دانشکده مهندسی برق و کامپیوتر – تهران - ایران
b.alizadeh@ut.ac.ir

چکیده: همچنانکه که بر پیچیدگی طراحی‌های توان پایین افزوده می‌شود، ابزارهای خودکارِ کارآمدتری به منظور درستی‌سنجی عملکرد آن‌ها مورد نیاز است. درستی‌سنجی همزمان عملکرد طراحی‌ها و سازگاری بخش کنترلی مدیریت توان با هدف توان پایین آن‌ها یکی از چالش‌های بزرگ است. این مقاله روشی ارائه می‌دهد که این مشکل را در پردازنده‌های مدرن توان پایین پیچیده که دارای ده‌ها حوزه توانی هستند، حل نماید. برای اطمینان از این که عملکرد پردازنده پس از قرار گرفتن بخش کنترل مدیریت توان تغییر نمی‌کند، بررسی برابری کارآمدی بین مدل پیاده‌سازی توان پایین و مدل مشخصه آن انجام می‌شود. با این حال، این نوع درستی‌سنجی به دلیل رفتار غیرعملکردی استراتژی‌های مدیریت توان در سطح سیستم کافی نیست. بنابراین، روش پیشنهادی سازگاری بین Power Management Unit (PMU) و Unified Power Format (UPF) را به وسیله قوانین توانی سطح بالای استخراج شده از UPF بررسی می‌کند. نتایج تجربی نشان می‌دهد که روش پیشنهادی نه تنها به طراحان کمک می‌کند تا یک کنترل‌کننده مدیریت توان سطح بالای صحیح بسازند بلکه همچنین بتوانند ایرادهای عملکردی توان پایین در طراحی‌شان را شناسایی کنند.

واژه‌های کلیدی: درستی‌سنجی صوری، معماری مدیریت توان، سطح سیستم، پردازنده توان پایین.

نوع مقاله: پژوهشی

DOI: 10.29252/jiaeee.18.4.203

تاریخ ارسال مقاله: ۱۳۹۸/۴/۱
تاریخ پذیرش مشروط مقاله: ۱۳۹۸/۱۱/۰۵
تاریخ پذیرش مقاله: ۱۳۹۹/۴/۳۰
نام نویسنده‌ی مسئول: دکتر بیژن علیزاده
نشانی نویسنده‌ی مسئول: تهران، خیابان کارگر شمالی، پردیس دانشکده‌های فنی دانشگاه تهران، دانشکده مهندسی برق و کامپیوتر

۱- مقدمه

برای رسیدن به حداکثر صرفه‌جویی در توان، استراتژی‌های مدیریت توان[۱] در سطح سیستم[۲] به عنوان نقطه شروع طراحی مورد استفاده قرار گرفته است. با این حال، ایجاد استراتژی سطح بالای صحیح نیازمند در نظر گرفتن هدف توان پایین[۳] و همچنین عملکرد مناسب طراحی پس از قرار دادن آن است. اگر چه، قالب توانی متحد[۴] [۱] امکان تعریف و درستی‌سنجی هدف توانی در سطح ثبات[۵] را می‌دهد، درستی‌سنجی بخش کنترلی توان در سطح سیستم همچنان یک چالش بزرگ محسوب می‌شود.

در این زمینه مقالات متعددی وجود دارند که به درستی‌سنجی طراحی‌های توان پایین از سطح گیت[۶] تا سطح تراکنش[۷] (TL) می‌پردازند [۲] - [۱۲]. در این مقالات تنها به یکی از دو هدف درستی‌سنجی بخش کنترلی مدیریت توان یا درستی‌سنجی عملکرد طراحی توان پایین پرداخته می‌شود و صحت توامان این دو هدف مورد توجه و بررسی قرار نگرفته است. در [۲] و [۳]، چارچوب‌هایی برای مدلسازی مدیریت توان در سطح سیستم الکترونیکی[۸] (ESL) ارائه شده است که مفاهیم UPF را به سطح TL و SystemC نگاشت می‌دهد.

در [۴] یک روش درستی‌سنجی صوری برای کنترل‌کننده‌های مدیریت توان سطح بالا با استفاده از ادعاهای توان پایینی که به طور دستی مشخص شده‌اند، پیشنهاد شده است. در یک روش مشابه [۵]، استراتژی‌های مدیریت توان که به صورت ترکیبی از سخت افزار و نرم-افزاری هستند، درستی‌سنجی شده است. این نوع از درستی‌سنجی‌ها برای یک سیستم کاملاً ضروری است، اما باید اطمینان حاصل کنیم که صحت عملکرد طراحی‌ها و هدف توان پایین (غیر عملکردی) آن‌ها به طور همزمان حفظ می‌شوند.

در [۶] روشی پیشنهاد شده است که حوزه‌های توانی[۹] معادل را به منظور افزایش سرعت درستی‌سنجی ترکیب می‌کند. در [۷] مسیرهای حوزه‌های توانی متقابل[۱۰] در طراحی فیزیکی بر اساس قوانین الکتریکی و ولتاژ پایدار درستی‌سنجی شده اند. در [۸] و [۹] روش‌های درستی‌سنجی گیت‌گذاری توان برای پردازنده‌ها پیشنهاد شده است که در آن از روش بررسی هم ارزی به منظور بررسی عملکرد مدل پیاده‌سازی بر اساس مدل مشخصه[۱۱] بدون ویژگی‌های توان مورد استفاده قرار می‌گیرد. در این روش، پردازنده‌ها به گونه‌ای چکیده شده‌اند که تمام اجزای آن در یک و یا دو حوزه‌ی توانی (حوزه‌های توانی با گیت‌گذاری توانی و بدون گیت‌گذاری توان) قرار می‌گیرند. روش [۹] به صورت دستی سیگنال‌های کنترل توان را خارج از مدل پیاده‌سازی به منظور درستی‌سنجی بخش کنترلی توان تنظیم می‌کند. اگر چه این روش‌ها مفید هستند، اما نمی‌توانند به بررسی جامع بخش‌های مدیریت توانی که دارای ده‌ها حوزه‌ی توانی و حالت‌های توانی مختلف هستند بپردازند. در [۱۰] واحد مدیریت توان[۱۲] (PMU) به طور خودکار از UPF و بخش مدیریت توان سراسری (GPM) استخراج شده و سپس به بخش مدل پیاده‌سازی چکیده افزوده می‌شود. به منظور اطمینان از اینکه مدل پیاده‌سازی پس از افزودن PMU به آن کار می‌کند، هم ارزی بین مدل مشخصه و مدل پیاده‌سازی جدید بررسی می‌شود. در [۱۱] یک روش درستی‌سنجی مبتنی بر ادعا برای بررسی سیستم یکپارچه با سیگنال‌های مختلط آنالوگ-دیجیتال (AMS-SoC) پیشنهاد شده است. در [۱۲] یک مشخصه‌ی توانی در سطح سیستم ارائه شده است که شامل هدف توانی چکیده شده و کنترل‌کننده مدیریت توان سطح بالا است. این مشخصه توانی به مدیریت توانی در سطح رجیستر و هدف توانی در UPF سنتز می شود. کنترل‌کننده مدیریت توان در سطح رجیستر تولید شده با استفاده از ادعاهای موجود در هر حوزه (همانند توالی‌های غیر قانونی سیگنال‌های کنترل توان) درستی‌سنجی می‌شود. در نهایت، در [۱۳] یک روش صوری برای اصلاح معماری مدیریت توان در پردازنده توان پایین با استفاده از مکانیسم SAT افزایشی ارائه شده است.

به تازگی، چندین ابزار همانند JasperGold [۱۴]، MVSIM و MVRC [۱۵] برای انجام درستی‌سنجی مدیریت توان توسعه یافته‌اند. در این ابزار قابلیت‌های شبیه‌سازی توان آگاه، بررسی صحت اتصالات سیگنال‌ها بین حوزه‌های توانی مختلف، تشخیص هر گونه گم شدن سیگنال‌ها و سلول‌های خاص و درستی‌سنجی ترتیب درست روشن شدن / خاموش شدن یک ماژول توسط بررسی ادعاهای استخراج شده از UPF وجود دارد.

با این حال، هیچ کدام از موارد ذکر شده در [۲] - [۱۵] نمی‌توانند به درستی‌سنجی کنترل‌کننده مدیریت توان سطح بالا با توجه به هدف توانی تعریف شده در UPF که شامل قوانین هر حوزه‌های توانی و همچنین قوانین بین حوزه‌های توانی است بپردازد، در حالی که عملکرد صحیح طراحی‌های توان پایین نیز حفظ شود. در این مقاله یک روش خودکار برای درستی‌سنجی پردازنده‌های توان پایین که شامل چندین حوزه توانی و استراتژی‌های مختلف مدیریت توان در سطح سیستم است، ارائه می‌شود. در روش پیشنهادی، ابتدا اطمینان حاصل می‌شود که طراحی پس از قرار دادن بخش‌های مدیریت توان همچنان درست کار می‌کند (بخش ۶). مرجع طلایی برای این هدف یک مدل مشخصه فاقد ویژگی‌های توانی است، در حالی که مدل پیاده سازی خط لوله‌ی پردازنده دارای چندین استراتژی مدیریت توان است. پس از این مرحله، درستی‌سنجی غیر عملکردی با استفاده از قوانین سطح بالا به طور خودکار استخراج شده از UPF 3.0 انجام می‌شود (بخش ۷). توجه داشته باشید که بر خلاف روش‌های [۴] و [۱۲]، روش پیشنهادی، سازگاری دو بخش مدیریت توان (GPM / LPM و UPF) که در سطوح مختلفی قرار دارند بررسی می‌کند. از این رو، نوآوری‌های روش پیشنهادی در این مقاله به شرح ذیل است:

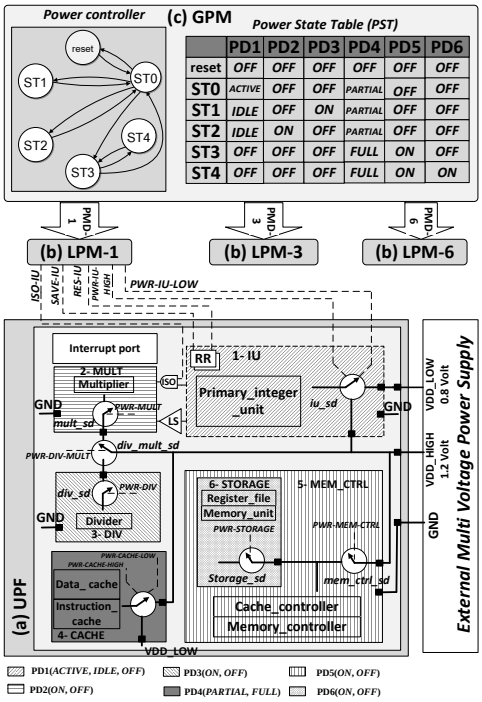
• ارائه یک روش جدید برای درستی‌سنجی صوری ویژگی‌های غیرعملکردی در مشخصه‌های توان پایین به منظور تضمین سازگاری بین UPF و PMU (بخش ۷).
• معرفی یک روش کارآمد برای درستی‌سنجی عملکرد پردازنده‌ها پس از افزودن استراتژی‌های توان پایین به آن‌ها (بخش ۶).
• مدل‌سازی و تولید واحد مدیریت توان (PMU) با استفاده از استخراج ویژگی‌های توانی موجود در UPF و GPM/LPM (بخش ۴ و بخش ۵).

مابقی مقاله به شرح ذیل سازماندهی شده است. در بخش ۲ یک مثال انگیزشی ارائه می‌گردد. در بخش ۳ مروری بر روش درستی‌سنجی پیشنهادی برای پردازنده‌های توان پایین می‌شود. بخش ۴ نحوه تولید PMU از GPM / LPM را نشان می‌دهد. در بخش ۵، یک مدل توان پایین چکیده شده ارائه می‌شود. در بخش ۶ عملکرد پردازنده‌ها بعد از افزودن PMU به آن‌ها بررسی می‌گردد. در بخش ۷ سازگاری PMU با هدف توانی موجود در UPF بررسی می‌شود. بخش ۸ نتایج تجربی را نشان می‌دهد و در نهایت، بخش ۹ نتیجه‌گیری و کارهای آتی آورده می‌شود.

۲- مثال انگیزشی

به منظور روشن شدن هدف اصلی درستی‌سنجی پردازنده‌های توان پایین با کنترل‌کننده‌های مدیریت توان در سطح سیستم، اجازه دهید که بخش مدیریت پردازنده LEON3 را همانند شکل (۱) در نظر بگیریم. این پردازنده دارای نه مولفه‌ی توانی اصلی به نام‌های ۱) Primary_integer_unit که شامل بخش‌هایی همچون ALU، Take branch و غیره است، ۲) Instruction_cache، ۳) Data_cache، ۴) Register_file، ۵) Multiplier، ۶) Divider، ۷) Memory_unit، ۸) Memory_controller و ۹) Cache_controller، است.

در پایین‌ترین سطح پروتکل مدیریت توان، UPF حوزه‌های توانی را با مجموعه‌ای از مولفه‌های اشاره شده که همواره فرکانس و ولتاژ یکسانی دارند می‌سازد. در این مثال، حوزه‌های توانی عبارتند از: ۱) IU، ۲) MULT، ۳) DIV، ۴) STORAGE، ۵) CACHE و ۶) MEM_CTRL. همانطوری که در شکل (۱)(۱) نیز مشخص است، دو مولفه‌ی Instruction_cache و Data_cache در حوزه‌ی توانی همواره روشن CACHE قرار دارند در حالی که Primery_integer_unit در حوزه‌ی توانی IU قرار گرفته است.



شکل (۱): پروتکل مدیریت توان برای پردازنده‌ی LEON3 ۱) UPF، ۲) LPMs و ۳) GPM.

از طرف دیگر، UPF شبکه‌های تغذیه و مدارهای کنترل توان همانند سلول‌های جداسازی[۱۳]، سوییچ‌های توانی، تغییردهنده‌های سطح[۱۴] و نگه‌داری حالت را برای هر حوزه‌ی توانی مشخص می‌نماید. این قالب توانی متحد درگاه‌های تغذیه‌ی هر یک از حوزه‌های توانی را به منابع تغذیه خارجی از مناسب طریق نت‌ها متصل می‌کند. سوییچ‌های تغذیه بین نت‌ها و درگاه‌های تغذیه‌ی حوزه‌های توانی که دارای چندین مد توانی می‌باشند قرار می‌گیرند. به عنوان مثال، سوییچ iu_sd امکان اتصال به منابع ۱/۲ ولت و ۰/۸ ولت را برای حوزه‌ی توانی IU فراهم می‌کند. UPF ساختار سلسله مراتبی را برای حوزه‌های توانی با استفاده از ترتیب سوییچ‌ها مشخص می‌کند. به عنوان مثال، storage_sd رتبه‌ی پایین‌تری نسبت به mem_ctrl_sd دارد، در نتیجه حوزه‌ی توانی STORAGE یک زیر حوزه‌ی توانی برای حوزه‌ی توانی MEM_CTRL محسوب می‌شود. این استاندارد وضعیت حوزه-های توانی را نسبت به یکدیگر مشخص می‌کند. به عنوان مثال سوییچ div_mult_sd به هر یک از دو سوییچ div_sd و mult_sd متصل است و اجازه می‌دهد که تنها یکی از این دو سوییچ فعال شوند. بنابراین یکی از دو حوزه‌ی توانی DIV و MULT می‌توانند در مد توانی روشن قرار داشته باشد. سلول جداساز بین دو حوزه‌ی توانی (همانند IU و MULT) می‌توانند سیگنال‌های مشترک آن‌ها را هنگامی که یکی از آن‌ها خاموش است قطع کند. سلول نگه‌داری برای حوزه‌های توانی که مقدار خود را قبل از خاموشی حفظ می‌کنند، قرار داده می‌شود. در سطح بالاتر از پروتکل مدیریت توان، مدیریت توانی محلی[۱۵] یا LPMها قرار گرفته‌اند که سیگنال‌ها کنترل توان سلول‌های

توانی (همانند ISO-IU، PWR-IU، RES-IU و SAVE-IU) تعریف شده در UPF را تنظیم و ترتیب فعال آن را مشخص می‌نمایند تا گذار بین مدهای توانی[16] حوزه‌ها امکان پذیر شود. توجه شود که مد توانی حالت عملکردی از یک حوزه‌ی توانی است که به وسیله‌ی یک زوج ولتاژ و فرکانس تعریف می‌شود. در بالاترین سطح پروتکل مدیریت توان، مدیریت توان سراسری[17] استراتژی‌های توانی را بر اساس کاربرد طراحی مشخص می‌نماید. این بخش از مدیریت توان بین حالت‌های توانی مجاز[18] گذار کرده و وضعیت هر یک از حوزه‌های توانی را مشخص می‌نماید. در این مثال GPM دارای شش حالت توانی reset، ST0، ST1، ST2، ST3، ST4 است. توجه شود که هر حالت توانی ترکیبی از مدهای توانی مربوط به همه حوزه‌های توانی است. به عنوان مثال، در حالت توانی ST0، مدهای توانی IU، MULT، DIV، CACHE، MEM_CTRL و STORAGE به ترتیب عبارتند از ACTIVE، OFF، PARTIAL، OFF و OFF.

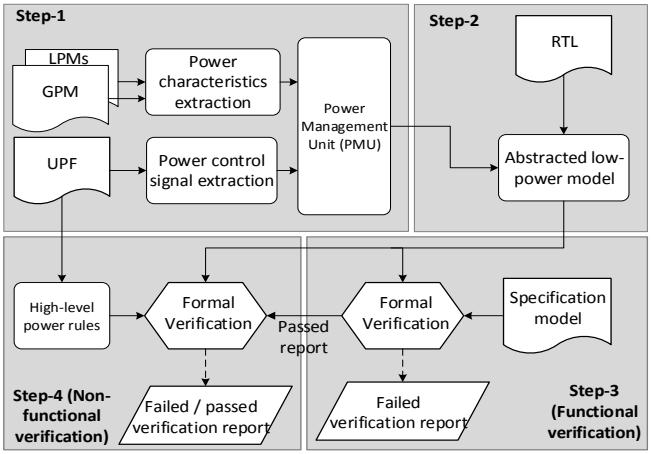
توجه شود که UPF به عنوان مشخصه‌ی توانی سطح پایین در نظر گرفته می‌شود و دو بخش دیگر مدیریت توان (GPM و LPMs) می‌بایست سازگاری کاملی با آن داشته باشند. به عنوان نمونه، در این مثال GPM نباید شامل حالت توانی باشد که دو حوزه‌ی توانی MULT و DIV در آن به طور همزمان روشن هستند. بنابراین، در این مقاله هدف از درستی‌سنجی پردازنده‌های توان پایین تنها بررسی هم ارزی این مدل‌ها با مدل مشخصه فاقد ویژگی‌ها توانی نبوده بلکه به صحت تطابق بخش‌های مختلف مدیریت توان با قوانین و هدف‌های توانی نهادینه شده در UPF نیز پرداخته می‌شود.

۳- مروری بر روش پیشنهادی

ورودی‌های روش درستی‌سنجی پیشنهاد شده عبارتند از مشخصه‌ی توانی در UPF، یک توصیف سطح بالا از طراحی به عنوان مدل مشخصه‌ی طراحی فاقد ویژگی‌های توان (SPEC) و مدل پیاده‌سازی توان پایین[19]. مدل پیاده‌سازی توان پایین نیز از سه جزء اصلی تشکیل شده است: GPM، LPMs و RTL. شکل (۲) مراحل اصلی روش پیشنهاد شده را نشان می‌دهد که به طور خلاصه عبارتند از:

- **مرحله‌ی اول:** استخراج ویژگی‌های توانی به منظور ساختن واحد مدیریت توان (PMU) از GPM و LPMها. در بخش ۴ این مرحله تشریح می‌شود.
- **مرحله‌ی دوم**: به منظور ساده‌سازی فرآیند درستی‌سنجی پردازنده‌ها، مدل توان پایین چکیده‌شده‌ای از پردازنده‌ها ارائه می‌شود که با خلاصه کردن مولفه‌های پردازنده و سپس افزودن واحد مدیریت توان به آن بدست می‌آید. این مرحله در بخش ۵ توضیح داده می‌شود.
- **مرحله‌ی سوم**: در این مرحله به درستی‌سنجی مبتنی بر بررسی برابری میان مدل SPEC از پردازنده و مدل توان پایین چکیده پرداخته می‌شود. اگر چه این هم ارزی مدل مشخصه با مدل توان پایین از جمله شروط لازم برای صحت پردازنده توان پایین است اما شرط کافی نیست. این مرحله در بخش ۶ ارائه می‌گردد.
- **مرحله‌ی چهارم**: پس از درستی‌سنجی عملکرد طراحی در مرحله‌ی سوم، با استفاده از گزاره‌های استخراج شده از UPF و تبدیل آن‌ها به قوانین توانی سطح بالا به بررسی صحت واحدهای مدیریت توانی و مطابقت آن‌ها با محدودیت‌های سطح پایین پرداخته شده است. این مرحله در بخش ۷ آورده شده است.

در مرحله ۳ و ۴ روش پیشنهادی که مراحل درستی‌سنجی مقدماتی و نهایی هستند از ابزار UCLID 3.1 استفاده شده است. توجه شود که روش پیشنهادی ارائه شده در این مقاله مستقل از قالب نرم‌افزاری است. بنابراین، می‌تواند بر روی هر ابزار درستی‌سنجی صوری با شبیه‌سازی نمادین و روند تصمیم گیری اجرا شود. در بخش‌های بعدی به تشریح هر یک از این مراحل می‌پردازیم.



شکل (۲): نمای کلی روش درستی‌سنجی توان پایین پیشنهادی

۴- مرحله‌ی اول: تولید واحد مدیریت توان

به منظور تولید PMU، ابتدا روش پیشنهادی ویژگی‌های توان را از GPM / LPM استخراج می‌کند (بخش ۴-۱) و سپس سیگنال‌های کنترل توان و ترتیب آن‌ها از UPF استخراج می‌گردند (بخش ۴-۲).

۴-۱- استخراج ویژگی‌های توانی[20] از GPM و LPMها

مدیریت توان سراسری و مدیریت توانی محلی که بیان کننده‌ی استراتژی‌های اصلی مدارهای توان پایین هستند در روش پیشنهادی به عنوان ورودی در نظر گرفته شده‌اند. این استراتژی‌ها معمولاً با استفاده از ماشین حالت محدود و به صورت ترکیبی از زبان نرم‌افزار و سخت‌افزار بیان می‌شوند. در این مرحله GPM و LPMها به صورت خودکار تجزیه و تحلیل شده و سپس به استخراج یکسری ویژگی‌های توانی از آن‌ها پرداخته می‌شود. این ویژگی‌های توانی شامل وضعیت

هریک از حوزه‌های توانی در هر حالت توانی، شرط گذار از یک حالت توانی به حالت توانی دیگر، حالت توانی فعلی، تعداد حوزه‌های توانی خاموش و روشن و همچنین مقدار سیگنال‌های کنترل توان در هر حالت توانی می‌باشند. این مرحله از کار شامل سه فاز اصلی است که عبارتند از: ۱) تشخیص حالت توانی فعلی و بعدی، ۲) تشخیص اثر گذارهای توانی بر روی هر LPM و ۳) مقدار سیگنال‌های کنترل توان هر LPM.

بخشی از مدیریت توان سراسری مربوط به پردازنده‌ی LEON3 در شکل (۳) به زبان SystemC نشان داده شده است. همانطور که در این شکل نیز مشخص است، حالت‌های توانی برای این پردازنده عبارتند از reset، ST0، ST1، ST2، ST3 و ST4 (خط ۴). همچنین GPM شرایط گذار از ST3 به ST4 را همانند خط‌های ۱۶، ۱۸ و ۲۵ تعیین می‌نماید. به عنوان مثال، در حالت ST3، اگر *empty_cache & finish_iu & ready_storage* صحیح باشد، سپس GPM یک گذاری از ST3 به ST4 خواهد داشت (خط ۲۱). همچنین اگر *finish_mem_ctrl & ready_iu* صحیح باشد، سپس گذار حالت توانی از ST3 به ST0 رخ می‌دهد (خط ۲۸) و در غیر اینصورت در حالت توانی ST3 باقی می‌ماند (خط ۳۶). همچنین شکل (۴)، بخشی از کد سخت افزاری LPM مربوط به حوزه‌ی توانی *STORAGE* را نشان می‌دهد (LPM-6). در مرحله‌ی اول، برای هریک از سیگنال‌های کنترل توانی این حوزه توانی مقدار پیش فرضی در نظر گرفته شده است (خطوط ۳ تا ۶). سپس، بسته به این که حالت توانی فعلی و قبلی به چه صورت است مقدار این سیگنال‌های کنترلی تغییر می‌نماید. البته توجه شود که ترتیب تغییر این سیگنال‌های کنترلی نیز توسط مدیریت توان محلی مشخص می‌شود. به عنوان مثال، هانطوری که در این شکل نیز مشخص است، اگر حالت توانی فعلی از ST3 به ST4 تغییر وضعیت دهد، آنگاه سیگنال‌های کنترل توانی مربوط به STORAGE به ترتیب به صورت "*PWR-STORAGE* = 1"، "*ISO-STORAGE* = 0" و "*RES-STORAGE* = 1" تغییر می‌نمایند.

شکل ۵، فرآیند استخراج ویژگی‌های سطح بالا را از مدیریت توان محلی و مدیریت توان سراسری نشان می‌دهد. در فاز اول حالت توانی فعلی و بعدی، در فاز دوم مد توانی تغییر یافته در حالت توانی بعدی و در فاز سوم مقدار سیگنال‌های کنترل توان و ترتیب آن‌ها استخراج می‌شود. در این شکل ترتیب سیگنال‌ها از چپ به راست می‌باشند.

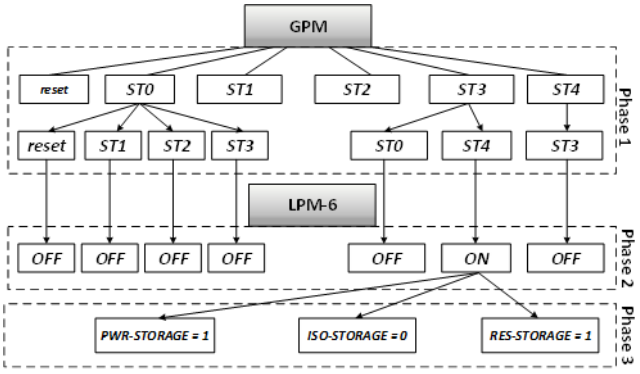
در شکل (۶)، ترتیب و مقدار سیگنال‌های کنترل توان استخراج شده از GPM/LPM به صورت یک ماشین حالت[۲۱] نشان داده شده است. توجه شود که این مشخصه‌های بدست آمده ممکن است درست و یا نادرست باشند که می‌بایست در بخش‌های بعدی مورد بررسی قرار گیرند.

```
1.  SC_MODULE (GPM_LEON3)
2.  {
3.  sc_in <sc_logic> clk, rst; // Inputs of GPM
4.  enum STATE {reset, ST0, ST1, ST2, ST3, ST4};
5.  sc_out enum next_state, present_state;   // Output of GPM
6.  SC_CTOR (GPM_LEON3)
7.  {
8.  SC_THREAD (GPM_LEON3Func);
9.  sensitive << clk.pos(); // sensitive list
10. }
11. void GPM_LEON3Func(); // calling void function
12. }; // end module
13. void GPM_LEON3::GPM_LEON3Func()
14. {
15. STATE curSt;
                    •••
16. else if (curSt.read() == ST3) // IDLE_iu, ON_mem_ctrl
17. {
18. if ((empty_cache.read() && finish_iu.read()) &&
    (ready_storage_elem.read())
19. {
20.         wait(delay);
21.         curSt = ST4; // Transition :: ST3 – ST4'
22.         next_state = curSt;
23.         present_state = ST3;
24. }
25. else if (finish_mem_ctrl.read() && ready_iu.read())
26. {
27.         wait(delay);
28.         curSt = ST0; // Transition :: ST3 – ST0'
29.         next_state = curSt;
30.         present_state = ST3;
31. else
32. {
33.         wait(delay);
34.         curSt = ST3; // Transition :: ST3 – ST3'
35.         next_state = curSt;
36.         present_state = ST3;} } // end void function
```

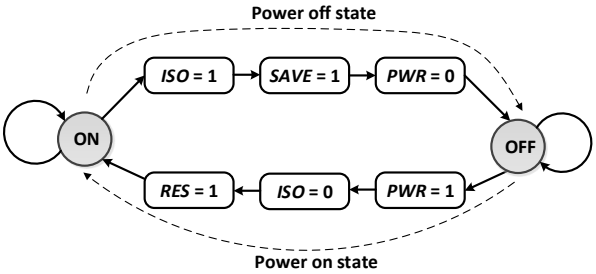
شکل (۳): بخشی از GPM پردازنده‌ی LEON3 با پنج حالت توانی

```
1.  always @ (posedge clk or negedge reset) begin
2.  if (! reset) begin
3.  storage_pwr <= 1'b0;
4.  storage_iso <= 1'b1;
5.  storage_on_state <= STORAGE_OFF_ON;
6.  storage_off_state <= STOARGE_ON_OFF;
7.  end else begin
8.  case (storage_on_state)
9.  STORAGE_OFF_ON: begin
10. if (present_state <= ST3 & next_state == ST4) begin
11.         storage_on_state <= STOARGE_ON_SD;
12. end
13. end
14. STOARGE_ON_SD: begin
15.         PWR-STORAGE <= 1'b1;
16.         if(PWR_ACK_STOARGE) begin
17.         storage_on_state <= STORAGE_ON_ISO;
18.         end
19. end
20. STORAGE_ON_ISO: begin
21.         ISO-STORAGE <= 1'b0;
22.         storage_on_state <= STORAGE_ON_RES;
23. end
24. STORAGE_ON_RES: begin
25.         RES-STORAGE <= 1'b1;
26.         storage_on_state <= STORAGE_ON_DONE;
27. end
28. STOARGE_ON_DONE: begin end
29. endcase
                    •••
30.                 end
```

شکل (۴): بخشی از مدیریت توان محلی *STORAGE* (LPM-6)



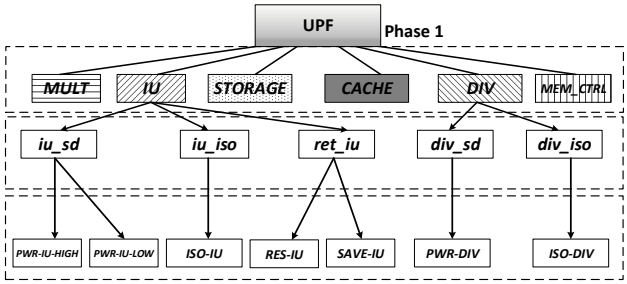
شکل (۵): مشخصه‌های توانی سطح بالایی که از GPM و LPM-6 استخراج شده است



شکل (۶): ماشین حالت توانی استخراج شده از LPM-6.

۴-۲- استخراج سیگنال‌های کنترل توان از UPF

قالب توانی متحد در پروتکل مدیریت توان در پایین‌ترین سطح قرار گرفته است و این ماژول تنظیم مدارهای کنترل توان سطح پایین را بر عهده دارد. در این مرحله از قالب توانی متحد یکسری اطلاعات مهمی استخراج می‌گردد که شامل تعداد حوزه‌های توانی، نام سلول‌ها و نام سیگنال‌های کنترل توانی این سلول‌ها است. سیگنال‌های کنترل توانی عبارتند از سیگنال‌های جداساز، سیگنال‌های ذخیره‌سازی، سیگنال بازیابی و سیگنال توان. به منظور استخراج سیگنال‌های کنترل توانی برای هر یک از حوزه‌های توانی می‌بایست روندی همانند شکل (۷) انجام شود. این شکل مراحل استخراج سیگنال‌های کنترل توانی و وضعیت آن‌ها را به ازای هر مد توانی برای پردازنده‌ی LEON3 نشان می‌دهد. روند استخراج شامل ۴ مرحله است. در مرحله‌ی دوم می‌بایست نام تمام حوزه‌های توانی مختلف استخراج شود که عبارتند از *IU*، *MULT*، *DIV*، *STORAGE*، *CACHE* و *MEM_CTRL*. در مرحله سوم نام مدارهای کنترل توان هر یک از حوزه‌های توانی مشخص می‌گردد. به عنوان مثال، حوزه‌ی توانی IU دارای سه مدار به نام‌های *iu_sd*، *iu_iso* و *iu_ret* است که به ترتیب مربوط به سوییچ توان، سلول جداساز و سلول نگه‌دارنده است. در مرحله‌ی چهارم، نام سیگنال‌های کنترلی برای هر یک این سلول‌ها مشخص می‌شود. به عنوان مثال سیگنال‌های کنترلی مربوط به حوزه‌ی توانی *IU* عبارتند از ISO-IU، PWR-IU، SAVE-IU و RES-IU که به ترتیب مربوط به سیگنال‌های کنترلی جداساز، توان، ذخیره‌سازی و بازیابی هستند.



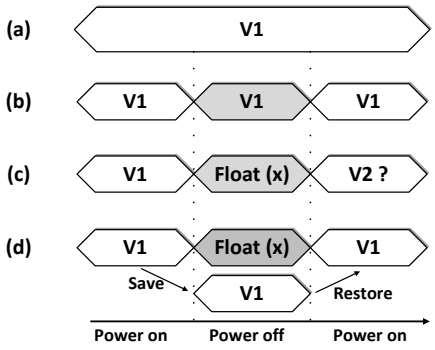
شکل (۷): سیگنال‌های کنترل توان استخراج شده از مدل UPF 3.0 پردازنده LEON3.

۵- مرحله‌ی دوم: تولید مدل توان پایین چکیده[۲۲]

در این مقاله به منظور کاهش پیچیدگی درستی‌سنجی در پردازنده‌های توان پایین، روشی را ارایه کرده‌ایم که پردازنده‌ها را به قالب چکیده‌ای مدل می‌کند. مدل توان پایین چکیده شامل دو قسمت ۱) مسیر داده چکیده شده۲) واحد مدیریت توان (PMU) است. در قسمت اول جزییات مولفه‌های مسیر داده حذف می‌شود، تعداد دستورالعمل‌ها کاهش یافته و به چند دستورالعمل اصلی دسته‌بندی می‌شوند. همچنین، مولفه‌های مسیر داده[۲۳] بر حسب ویژگی‌های توان پایین آن‌ها به چهار گروه تقسیم‌بندی می‌شوند در حالی که بخش کنترلر آن به صورت دست نخورده باقی می‌ماند. این ویژگی‌های توانی شامل نوع مدهای توانی، مقادیر خروجی مولفه‌ها در مد توانی خاموش و چگونگی ذخیره‌سازی و بازیابی اطلاعات در گذار از خاموش به روشن و بالعکس است. چهار گروه واحدهای مسیر داده عبارتند از:

- **همواره روشن**[۲۴]: این دسته همیشه در مدهای توانی روشن قرار دارند و مقادیر خروجی مشخصی در طی فرآیند روشن شدن و خاموش شدن پردازنده دارند. همانند شکل (۸)-الف.
- **نگه‌داری مقادیر در خودش**[۲۵]: این دسته حالت‌ها و مقادیر خودشان را در هنگام فرآیند خاموش شدن در ثبات‌های نگه‌داری حفظ می‌کنند. همانند شکل (۸)-ب.
- **از دست دادن مقدار**[۲۶]: این دسته مقادیر خود را به محض ورود به مد توانی خاموش از دست می‌دهند و پس از ورود به مد توانی روشن دارای مقدار و حالت نامشخص (X) هستند. همانند شکل (۸)-ج.
- **نگه‌داری مقدار در حافظه‌ی خارجی**[۲۷]: این دسته حالت‌ها و مقادیر قبل از اینکه به مد خاموشی بروند در حافظه‌های خارجی ذخیره کرده و در طی فرآیند روشن شدن از آن‌ها بازیابی می‌کنند. همانند شکل (۸)-د.

به عنوان مثال در پردازنده‌ی LEON3، به منظور کاهش پیچیدگی مسیر داده، مولفه‌های Instruction_cache، Data_cache، Multiplier، Divider، Memory_ctrl و ALU با استفاده از توابع غیرتفسیری[۲۸] مدل‌سازی شده‌اند در حالی که مولفه‌هایی همانند Rigester_file و Memory_unit که نیاز به جزئیات بیشتر دارند با استفاده از توابع تفسیری[۲۹] مدل سازی شده‌اند. همچنین، مولفه‌هایی که خروجی آن‌ها از نوع بیت است همانند Take branch و Take trap با استفاده از توابع گزاره‌ای[۳۰] مدل‌سازی می‌شوند. در ضمن تمامی مقادیر سیگنال‌های کنترلی و داده در مسیر داده با استفاده از نمادهای TERM و TRUTH چکیده‌سازی شده‌اند. با این حال بخش کنترل-کننده مربوط به این پردازنده به صورت دست نخورده باقی مانده است. سپس، ۷۲ دستورالعمل مختلف در پردازنده‌ی LEON3 به ۷ دسته دستورالعمل کلی LD، ST، MUL، DIV، BR، TR و NULL تقسیم بندی می‌شود. سپس، مولفه‌های مختلف مسیر داده بر حسب مد توانی آن‌ها و سایر ویژگی‌های توانی ذکر شده به یکی از چهار گروه همواره روشن، از دست دادن مقدار، نگه‌داری مقدار در خود و نگه‌داری مقدار در حافظه خارجی تقسیم بندی می‌شوند. در پایان، واحد PMU به مدل پیاده‌سازی چکیده افزوده می‌شود تا مدل پیاده‌سازی توان پایین چکیده را بوجود آورد.



شکل (۸): دسته‌بندی مولفه‌های مسیر داده به چهار دسته، الف) همواره روشن، ب) نگه‌داری مقدار در خود، ج) از دست دادن مقدار، د) نگه‌داری مقدار در حافظه‌ی خارجی.

۶- مرحله‌ی سوم: بررسی برابری بین مدل توان پایین چکیده و مدل مشخصه

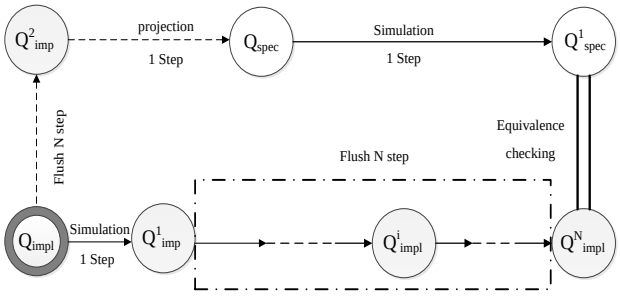
گام نخست روش درستی‌سنجی پیشنهادی اطمینان از صحت عملکرد پردازنده توان پایین چکیده پس از افزودن بخش مدیریت توان است. ورودی‌های این گام عبارتند از مدل مشخصه (مدل طلایی) که رفتار ترتیبی معماری مجموعه دستور العمل[۳۱] پردازنده را توصیف می‌کند و مدل پیاده سازی خط لوله[۳۲] که شامل پردازنده توان پایین چکیده بدست آمده از بخش ۵ است. در این گام، روشی مبتنی بر فلاش[۳۳] [۱۷] به منظور بررسی اینکه آیا حالت پردازنده‌ی خط لوله دارای استراتژی توان پایین با معماری مجموعه دستورالعمل فاقد ویژگی‌های توانی برابر است در ابزار UCLID 3.1 استفاده شده است. شکل (۹) روند گام اول برای درستی‌سنجی پردازنده‌ی توان پایین چکیده که شامل واحد PMU است را نشان می‌دهد. همانطوری که در این شکل نیز نشان داده شده است در ابتدا مدل پردازنده توان پایین چکیده با ورودی اختیاری برای یک سیکل شبیه‌سازی نمادین[۳۴] می‌گردد (Q_1^{impl}). سپس به منظور اینکه دستورالعمل‌های اجرا شده‌ی ناتمام موجود در خط لوله به طور کامل اجرا شوند و اثر این دستورات و دستور جدید بر روی قسمت‌های قابل مشاهده طراحی، همانند شمارنده برنامه (Q_N^{impl}.PC)، فایل ثبات (Q_N^{impl}.RF)، کش داده (Q_N^{impl}.dCache) و حافظه داده (Q_N^{impl}.dMem) دیده شود، خط لوله برای N سیکل فلاش می‌شود. تعداد فلاش‌های موجود برای کامل شدن دستورالعمل‌ها برابر با تعداد مراحل خط‌لوله است که به عنوان مثال برای پردازنده LEON3 برابر با هفت است.

به منظور داشتن مقایسه‌ای منصفانه بین مدل پیاده‌سازی و مدل مشخصه در گام بعد، ابتدا مدل پیاده‌سازی را بدون اینکه دستور جدیدی را وارد آن کنیم N بار فلاش می‌نماییم تا دستورات باقی مانده در خط لوله به طور کامل اجرا شوند و اثر خود را بر روی بخش‌های قابل مشاهده طراحی، یعنی شمارنده برنامه، فایل ثبات، کش داده و حافظه داده، بگذارند. سپس بخش‌های قابل مشاهده در مدل پیاده سازی را ذخیره سازی کرده (Q_1^{impl}) و در مدل مشخصه انتقال می‌دهیم ($Q_1^{impl} \rightarrow Q^{spec}$). آنگاه در مدل مشخصه به منظور اجرای دستورالعمل جدیدی که قبلا در مدل پیاده‌سازی اجرا شده بود، مدل مشخصه را یکبار با دستور العمل جدید شبیه‌سازی نمادین می‌نماییم. سپس بخش‌های قابل مشاهده در مدل مشخصه را ذخیره سازی می‌نماییم (Q^{spec}.PC, Q^{spec}.RF, Q^{spec}.dCache, Q^{spec}.dMem). در پایان به منظور درستی‌سنجی مدل مشخصه و مدل پیاده‌سازی رابطه‌ی (۱) را بررسی می‌کنیم که در آن بخش‌های قابل مشاهده در مدل مشخصه و مدل پیاده‌سازی نظیر به نظیر مقایسه می‌گردند. روش پیشنهادی در این مرحله که به منظور درستی‌سنجی مدل پیاده‌سازی چکیده توان پایین ارایه شده است دارای تعداد مراحل شبیه‌سازی نمادین کمتری نسبت به [۹] و مدل پیاده سازی چکیده فاقد واحد PMU است.

$$(Q_N^{impl}.PC = Q_1^{spec}.PC) \;\&\; (Q_N^{impl}.RF = Q_1^{spec}.RF) \;\&\; (Q_N^{impl}.dCache = Q_1^{spec}.dCache) \;\&\; (Q_N^{impl}.dMem = Q_1^{spec}.dMem) \quad (۱)$$

به منظور درستی‌سنجی مدل پیاده‌سازی خط لوله بدون PMU با مدل مشخصه، از نمودار بررسی مطابقتی[۳۵] استفاده می‌شود که در آن سیگنال‌های کنترل توان هر یک از حوزه‌ها در هر مرحله به صورت دستی و خارج از مدل پیاده‌سازی تنظیم شود. اگر N، PS و M به ترتیب تعداد مراحل خط لوله، تعداد متوسط سیگنال‌های کنترل توان

و تعداد متوسط حوزه‌های توانی باشند، سپس، تعداد شبیه‌سازی نمادین برای این حالت برابر است با $4\times(PS\times M) + 2\times N + 3$. این مقدار وابستگی زیادی به تعداد حوزه‌های توانی و سیگنال‌های کنترل توان دارد. این وابستگی موجب می‌شود که این روند برای پردازنده‌های واقعی که تعداد حوزه‌های توانی آن ۲۰ تا ۵۰ است قابل مقیاس‌گذاری نباشد. این در حالی است که تعداد شبیه‌سازی‌های نمادین برای روش پیشنهادی با PMU برابر با $2\times N + 3$ است. این مقدار مستقل از تعداد حوزه‌های توانی و تعداد سیگنال‌های کنترل توان است. در روش پیشنهادی واحد PMU به صورت خودکار سیگنال‌های کنترل توان و مدهای توانی حوزه‌های توانی را تنظیم می‌نماید و در نتیجه تعداد شبیه‌سازی‌های توانی مستقل از تعداد حوزه‌های توانی است و روند درستی‌سنجی با ورودی‌های اختیاری امکان پذیر است.



شکل (۹): بررسی مطابقتی میان SPEC و مدل پیاده‌سازی با PMU.

۷- مرحله‌ی چهارم: درستی‌سنجی PMU

همانطورکه قبلا اشاره شد، اگر چه یکسان بودن عملکرد مدل توان پایین با مدل مشخصه از جمله شروط لازم برای صحت پردازنده توان پایین است، اما شرط کافی نیست. چون چنین سیستم‌هایی دارای ویژگی‌هایی هستند که نمی‌توانیم در عملکردشان آنها را بیان کنیم. به همین دلیل، به این ویژگی‌ها، ویژگی‌های غیرعملکردی گفته‌ایم. بنابراین، پس از پایان روش اول درستی‌سنجی، می‌بایست پردازنده‌ی توان پایین را از لحاظ محدودیت‌های اعمال شده در UPF مورد ارزیابی قرار داد. از این رو، روش پیشنهادی گزاره‌های هدف توان کلی را از استراتژی‌های توان پایینی که در UPF و جدول حالت توانی[۳۶] آن (PST) موجود است استخراج می‌نماید. سپس، این گزاره‌های سطح پایین به یکسری قوانین توانی سطح بالایی تبدیل می‌شوند که می‌توانند به سرعت خطاهای موجود در مدیریت‌های سطح بالا همانند GPM و LPM را شناسایی نمایند. در نهایت، این قوانین موجب می‌شوند که طراح به سمت طراحی یک بخش مدیریت توان صحیح راهنمایی شود. برخی از این قوانین توانی استخراج شده از UPF و PST شامل موارد زیر می‌باشند:

- حوزه‌های توانی همواره روشن در هیچ حالت توانی نبایستی خاموش باشند.
- حوزه‌های توانی که با یکدیگر ناسازگار هستند نباید همزمان روشن باشند.
- حوزه‌های توانی همدوس می‌بایست همزمان روشن شوند.
- گذارهای مدهای توانی غیر مجاز نباید رخ دهند.

توجه شود که هریک از قوانین اشاره شده در بالا خود شامل ده‌ها گزاره‌ی سطح پایین استخراج شده می‌باشند. این گزاره‌های سطح پایین عبارتند از گزاره‌های مربوط به ترتیب صحیح سیگنال‌های کنترل توان و مقدار صحیح آن‌ها در هر مد توانی. از طرف دیگر برای اینکه یک حوزه‌ی توانی از یک مد توانی به مد توانی دیگر گذار کند می‌بایست ترتیب صحیحی از این سیگنال‌های کنترلی رخ دهد. در غیر اینصورت ممکن است در عملکرد آن خللی ایجاد شود. این ترتیب در استاندارد IEEE به ازای هر حالت توانی مشخص شده است [۱]. به عنوان مثال، گزاره‌ی زیر مربوط به خاموش شدن حوزه‌ی توانی STORAGE است.

```
pd_stoarge_off ($):=
    ((~PWR-STOARGE [$]) & (~RES-STORAGE [$]) &
    (ISO-STOARGE [$]) & (~SAVE-STORAGE [$]));
```

که در آنجا، عملگر [$] یعنی مقدار سیگنال‌های کنترل توان در مرحله‌ی زمانی $ مدنظر است.

همچنین، گزاره‌ی استخراج شده‌ی زیر نشان می‌دهد که ترتیب صحیح سیگنال‌های کنترل توان برای حوزه‌ی توانی STORAGE در گذار از OFF_MODE به ON_MODE به چه صورت است.

```
pd_stoarge_off_on ($):=
  (((PWR-STOARGE [$] & ~RES-STORAGE [$] & ISO-
STOARGE [$])
=> (PWR-STOARGE [succ ($)] & RES-STORAGE [succ ($)]
& ISO-STOARGE [succ ($)]))
=> ((PWR-STOARGE [succ^2 ($)] & ~RES-STORAGE [succ^2
($)] & ~ISO-STOARGE [succ^2 ($)])));
```

که در آن عملگر => استنتاج و نماد succ تابع مابعد را نشان می‌دهد. این هدف‌های توانی استخراج شده به ساختن قوانین توانی سطح بالا کمک می‌کنند. در زیر بخش‌های زیر به توضیح هریک از این قوانین توانی و نحوه‌ی نمایش آن‌ها پرداخته می‌شود.

۷-۱- حوزه‌های توانی همواره روشن[۳۷]

شکل (۱۰) ، بخشی از جدول حالت توانی (PST) برای پردازنده‌ی LEON3 را نشان می‌دهد، که به صورت خودکار توسط ابزار *Design Compiler (DC)* تولید شده است. روش پیشنهادی تمامی قوانین توانی خود را از طریق استفاده از گزاره‌های تولید شده در بخش ۷ و این جدول بدست می‌آورد. به عنوان مثال، همانطوری در این جدول

نیز مشخص است، حوزه‌ی توانی *CACHE* همواره در مد توانی روشن خود (*FULL_MODE* یا *PARTIAL_MODE*) قرار گرفته است. بنابراین، این مد توانی می‌بایست همواره روشن بوده و در همه‌ی حالت‌های توانی مربوط به GPM نباید خاموش گردد. به منظور بررسی این قانون و اعمال آن به مدیریت توان سطح بالا قانون زیر نوشته می‌شود:

```
always_on_cache:= FORALL($) (~pd_cahce_on ($) & pd_cache_off ($));
```

که در آنجا، pd_cache_on و pd_cache_off به ترتیب گزاره‌های تولید شده برای روشن و خاموش شدن حوزه‌ی توانی *CACHE* است که از بخش ۷ بدست آمده‌اند.

```
## Creating Power State Table ##

create_pst pst –domains {IU MULT DIV CACHE MEM_CTRL STORAGE}
add_pst_state s0 -pst pst -state {OFF_MODE OFF_MODE PARTIAL_MODE OFF_MODE OFF_MODE OFF_MODE}
add_pst_state s1 -pst pst -state {ACTIVE_MODE OFF_MODE PARTIAL_MODE OFF_MODE OFF_MODE OFF_MODE}
add_pst_state s2 -pst pst –state {IDLE_MODE ON_MODE PARTIAL_MODE OFF_MODE OFF_MODE OFF_MODE}
                    •••
add_pst_state s34 -pst pst –state {OFF_MODE OFF_MODE ON_MODE
```

شکل (۱۰): بخشی از جدول حالت توانی برای پردازنده LEON3.

۷-۲- حوزه‌های توانی ناسازگار[۳۸]

برخی از حوزه‌های توانی نمی‌توانند همزمان با هم روشن شوند زیرا ممکن است روشن بودن همزمان آن‌ها موجب اختلال در عملکرد کل سیستم شده و یا اینکه سوییچ‌های تغذیه آن‌ها به صورت معکوس یکدیگر فعال شوند. این نوع از حوزه‌های توانی را اصطلاحاً ناسازگار می‌گوییم. در شکل (۱۰) نمونه‌ای از این حوزه‌ها که عبارتند از *MULT* و *DIV* را می‌توان یافت. قانون زیر به منظور اعمال این حالت به مدیریت توان سطح بالا و جلوگیری از خطا در آن تولید می‌شود:

```
conflict_mult_div: = FORALL ($) ((pd_mult_on ($) & pd_div_on ($));
```

۷-۳- حوزه‌های توانی سازگار[۳۹]

برخلاف حوزه‌های توانی ناسازگار، بعضی از حوزه‌های توانی تنها زمانی روشن می‌شوند و یا به حداکثر توان خود می‌رسند که قبلاً یکسری حوزه‌های توانی دیگری که اصطلاحاً پیش نیاز گفته می‌شوند فعال شوند. به مجموعه‌ی اینگونه حوزه‌های توانی، حوزه‌های توانی سازگار گفته می‌شود. به عنوان مثال، با تجزیه و تحلیل شکل (۱۰) می‌توان مجموعه‌ای از اینگونه حوزه‌ها را پیدا نمود که عبارتند از STORAGE و MEM_CTRL و از طرف دیگر CACHE و MEM_CTRL. به عنوان مثال، قانون زیر حوزه‌ی توانی STORAGE را ملزم می‌کند که بعد از روشن شدن MEM_CTRL فعال شود.

```
coherence_storage_mem_ctrl:=FORALL ($) ((pd_storage_off_on ($)  => (pd_mem_ctrl_on ($));
```

همچنین قانون زیر به منظور اعمال محدودیت بر روی حوزه‌ی توانی *MEM_CTRL* و اینکه نباید پیش از *STORAGE* خاموش شود به مدیریت توان سطح بالا اعمال می‌شود.

```
coherence_mem_ctrl_storage:=FORALL($)((pd_mem_ctrl_on_off ($)  => (pd_ storage _off ($));
```

۷-۴- حالت‌های توانی غیر مجاز[۴۰]

قالب توانی متحد گذارهای مجاز و غیر مجاز بین مدهای توانی هر حوزه‌ی توانی را مشخص می‌نماید. به عنوان مثال یک حوزه‌ی توانی که دارای مد توانی روشن، بی کار و خاموش است نمی‌تواند مستقیماً از مد توانی روشن به مد توانی خاموش گذار کند. گزاره‌های زیر همین مطلب را برای حوزه‌ی توانی *IU* نشان می‌دهند:

```
add_mode_transition -domain IU
-transition {turn_on -from IDLE_MODE -to OFF_MODE -legal}
-transition {illegal1 -from ACTIEV_MODE -to OFF_MODE -illegal}
```

به منظور جلوگیری از رخ داد اینگونه گذارها، روش پیشنهادی با استخراج یکسری قوانین سطح بالا از وارد شدن مدیریت توان سراسری به این حالت‌ها جلوگیری می‌کند.

```
illegal_iu_off_on_trans:=FORALL($)((pd_iu_off($))=>pd_iu_on ($));
```

۸- نتایج تجربی

در این بخش به منظور بررسی کارایی متدولوژی درستی‌سنجی پیشنهادی، چهار آزمایش ترتیب داده شده است که بر روی شش مورد مطالعه به نام‌های PowerTrans [۵]، MIPS [۱۸]، LEON3 [۱۹]و eLEON3 [۲۰] اجرا می‌شوند. جدول ۱ آمارهای طراحی‌های مورد مطالعه را نشان می‌دهد. تعداد خطوط کد و تعداد مراحل خط لوله طراحی به ترتیب در ستون دوم و سوم نشان داده شده‌اند. ستون چهارم تعداد حالت‌های توانی و ستون پنجم تعداد حوزه‌های توانی طراحی‌ها را نشان می‌دهند. تمامی آزمایش‌ها بر روی سیستمی با مشخصات پردازنده‌ی 2.4 Gig Intel Core i7، با ۶ گیگابایت حافظه‌ی اصلی بر روی سیستم عامل Linux 32 bit اجرا می‌شوند. از UCLID 3.1 [۱۶] به عنوان ابزار درستی‌سنجی استفاده شده است که در آنجا MiniSAT به عنوان SAT Solver به کار می‌رود.

در آزمایش اول، اثر بررسی هدف توان کلی بر روی بخش‌های مدیریت توانی سطح بالای هر یک از طراحی‌ها مورد ارزیابی قرار گرفته است. در آزمایش دوم ضرورت استفاده از واحد مدیریت توان در مدل پردازنده‌ی توان پایین چکیده تشریح شده است. در آزمایش سوم تاثیر تعداد حوزه‌های توانی بر کارایی متدولوژی درستی‌سنجی پیشنهادی آورده شده است. در آزمایش چهارم متدولوژی درستی‌سنجی پیشنهادی با سایر روش‌های موجود در این زمینه [۹] مقایسه می‌شود.

جدول (۱): جزییات طراحی برای چهار حالت مورد آزمون

Design name	# Lines of code	# Pipeline stages	# GPM states	# Power domains
PowerTrans	373	3	7	3
MIPS	8750	5	5	4
LEON3	10809	7	6	6
eLEON3	14305	7-superscalar	6	7

۸-۱- اثر بررسی هدف توان کلی

در این آزمایش به بررسی تاثیر روش اول درستی‌سنجی و آنالیز هدف توان کلی استخراج شده از UPF در هر یک از طراحی‌های توان پایین PowerTrans، MIPS، LEON3 و eLEON3 پرداخته می‌شود. جدول (۲) تعداد گزاره‌ها و قوانین توانی استخراج شده هر طرح را نشان می‌دهد. ستون‌های ۲ تا ۴ به ترتیب تعداد گزاره‌های مربوط به حالت‌های توانی روشن و خاموش، تعداد گزاره‌ای مربوط به بالارفتن یا پایین آمدن حالت‌های توانی و تعداد گزاره‌های مربوط به گذار میانی بین دو حالت توانی روشن را نشان می‌دهند. ستون پنجم، تعداد قوانین توانی سطح بالای استخراج شده را نشان می‌دهد. جدول (۳) اثر این قوانین توانی استخراج شده را بر روی کارایی متدولوژی پیشنهادی نشان می‌دهد. در ستون دوم این جدول تعداد قوانین توانی استخراج شده نشان داده شده است. ستون سوم زمان ساختن که عبارتست از استخراج گزاره‌های توان پایین و تبدیل آن به قوانین توانی سطح بالا است را نشان می‌دهد. ستون چهارم زمان درستی‌سنجی این قوانین توانی را با استفاده از ابزار UCLID 3.1 نشان می‌دهد. همانطوری که در این جدول‌ها نیز مشخص است، روش پیشنهادی قادر است محدودیت‌های قوانین توانی استخراج شده از UPF را بر روی طراحی-های توان پایین با چندین حوزه‌ی توان در مدت زمانی کوتاهی بررسی نماید.

جدول (۲): آمار گزاره‌ها و قوانین استخراج شده از UPF و PST

Design name	Number of predicates			Number of power rules
	Power on/off	Power up/down	Intermediate transitions	
PowerTrans	6	6	0	3
MIPS	10	12	4	7
LEON3	14	24	4	10
eLEON3	18	28	6	14

جدول (۳): نتایج گام اول درستی‌سنجی بر روی موارد آزمون.

Design name	# power rules	Formal verification time (in second)		
		Built-time	Run-time	Total
PTrans	3	2.15	1.72	3.87
MIPS	7	7.82	8.43	16.25
LEON3	10	9.72	13.29	23.01
eLEON3	14	15.08	27.13	42.21

۸-۲- سودمندی افزودن PMU

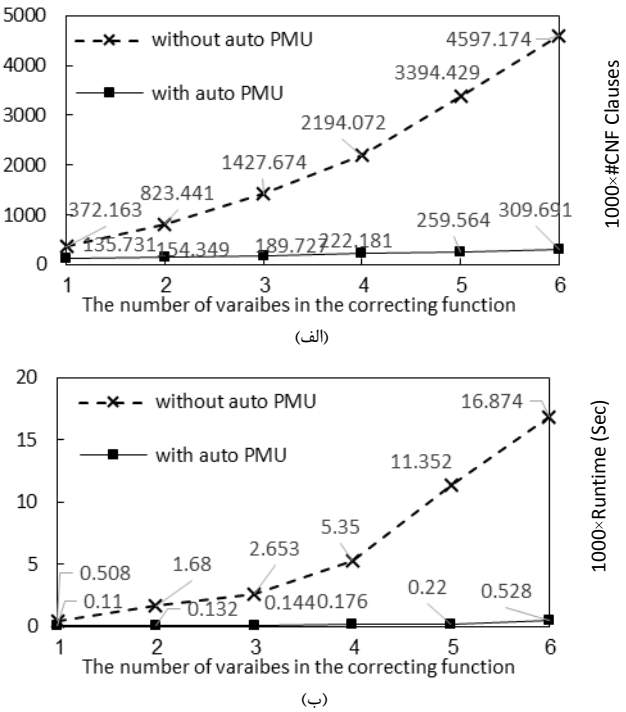
همانطوری که پیشتر نیز گفته شد، یکی از نوآوری‌های موجود در این مقاله تولید و افزودن خودکار PMU با استفاده از مشخصه‌های توانی استخراج شده از GPM/LPM و سیگنال‌های کنترل توان استخراج شده از UPF است. به منظور روشن شدن فواید این مولفه، کارایی روش پیشنهادی با استفاده از این مولفه و بدون آن بررسی می‌شود. جدول (۴) گزارشی از این نتایج را بر روی چهار مورد آزمون و در دو حالت با PMU و بدون PMU نشان می‌دهد. در این جدول Step#، Var#، Clause# و CPU Time به ترتیب تعداد مراحل شبیه‌سازی نمادین، تعداد متغیرهای CNF، تعداد عبارت‌های CNF و زمان درستی‌سنجی را نشان می‌دهند. همانطوری که در این جدول نیز نشان داده شده است زمان درستی‌سنجی با استفاده از شیوه‌ی مبتنی بر PMU تعداد مراحل شبیه‌سازی نمادین و زمان درستی‌سنجی را به ترتیب ۴/۸۵× و ۲۷/۶۸× در مقایسه با بدون PMU کاهش داده است.

جدول (۴): نتایج درستی‌سنجی بر روی دو حالت با PMU و بدون PMU

Design name	Without PMU				With PMU				Improvement (x)
	# Step	# Var	# Clause	CPU time (Sec)	# Step	# Var	# Clause	CPU time (Sec)	
PowerTrans	36	221973	785132	1151	9	25891	86765	79	×4
MIPS	45	276841	902241	1452	13	34178	100249	132	×3.46
LEON3	89	1659668	4597174	16874	17	104978	309691	528	×5.2
eLEON3	101	1883442	521697	19620	17	179686	397583	673	×5.9
Average	68	1010481	1701561	9774	14	86183	223572	353	×4.85

۸-۳- اثر تعداد حوزه‌های توانی

در این آزمایش، تاثیر افزایش تعداد حوزه های توانی را بر عملکرد روش پیشنهادی ۱) با PMU ۲) بدون PMU نشان داده شده است. شکل (۱۱)، تعداد CNFها و زمان درستی‌سنجی مربوط به این دو روش را برای مورد مطالعه LEON3 نشان می‌دهد. همانطور که در این شکل نیز مشخص است با افزایش تعداد حوزه‌های توانی تعداد CNFها و زمان درستی‌سنجی در روش بدون PMU به طور نمایی افزایش می‌یابد و این امر موجب می‌شود که این روش دارای قابلیت مقیاس گذاری مناسبی برای پردازنده‌هایی که دارای ۲۰ تا ۵۰ حوزه‌ی توانی هستند نباشد. در نتیجه، این روش کارایی لازم را برای درستی‌سنجی پردازنده‌های توان پایین مدرن را دارا نیست. این در حالی است که در روش پیشنهادی با PMU، تعداد CNF و زمان درستی‌سنجی به طور خطی و با شیب ملایم با افزایش تعداد حوزه‌های توانی زیاد می‌شود. این امر نشان دهنده‌ی این مطلب است که روش پیشنهادی با PMU تقریباً مستقل از تعداد حوزه‌های توانی می‌تواند به درستی‌سنجی پردازنده‌های توان پایین واقعی بپردازد.



شکل (۱۱): اثر افزایش تعداد حوزه‌های توانی بر روی: الف) تعداد عبارت‌های CNF. ب) برروی زمان درستی‌سنجی

۸-۴- مقایسه با روش‌های موجود

در پایان به منظور مقایسه روش پیشنهادی با سایر روش‌های موجود در زمینه‌ی درستی‌سنجی پردازنده‌های توان پایین، در این مرحله آزمایشی ترتیب داده شده است. در این آزمایش نتایج درستی‌سنجی روش پیشنهادی با روش نوین درستی‌سنجی مبتنی بر بررسی برابری ارائه شده در [۹] مورد مقایسه قرار گرفته است. در مقایسه با روش ارایه شده در [۹]، روش پیشنهادی با PMU، سیگنال‌های کنترل توان را به صورت خودکار و بر اساس استراتژی توانی تعریف شده برای هر حوزه‌ی توانی تنظیم می‌کند. سپس در اولین مرحله با استفاده از هدف توانی کلی به بررسی و آنالیز توانی مدل پیاده‌سازی توان پایین می‌پردازند. پس از طی شدن این مرحله، مدل پیاده‌سازی توان پایین که دارای ورودی‌های اختیاری است و مد توانی هر یک حوزه‌های توانی آن بر اساس واحد مدیریت توان کنترل می‌شود با مدل مشخصه که فاقد ویژگی‌های توانی است بررسی برابری می‌شود تا از لحاظ صحت عملکرد پس از افزودن استراتژهای توانی تصدیق شود. با توجه به اینکه زمان درستی‌سنجی در روش پیشنهادی با PMU وابستگی اندکی با تعداد حوزه‌های توانی دارد بنابراین دارای قابلیت مقیاس پذیری بالایی است. جدول (۵) نتایج درستی‌سنجی روش پیشنهادی با PMU و بدون PMU را برای مورد مطالعه‌ی MIPS در حالتی که دارای یک و سه حوزه‌ی توانی باشد را در مقایسه با روش ارایه شده در [۹] نشان می‌دهد. همانطوری که در جدول نیز مشخص است، روش پیشنهادی با PMU، با تعداد شبیه‌سازی‌های نمادین کمتر در مقایسه با [۹] تعداد CNF و زمان درستی‌سنجی را به ترتیب با ضریب ۱۷/۵۷× و ۱۳/۷۱× کاهش داده است.

جدول (۵): نتایج مقایسه روش پیشنهادی با [۹] بر روی MIPS

Method	#Domain	#Step	# CNF var	# CNF clause	CPU time (seconds)
Without PMU	1	25	277108	823441	132
	3	49	276841	902241	1452
Method in [9]	1	25	434837	1292158	924
	3	29	606104	1802971	1804
With PMU	1	13	25964	75898	67
	3	13	34178	100249	132
Improvement in comparison with		2.08×	11.04×	17.57×	13.71×

۹- نتیجه‌گیری

در این مقاله به تشریح روش درستی‌سنجی پیشنهادی که متشکل از بررسی قوانین سطح بالا بر روی بخش‌های مدیریت توان و همچنین بررسی برابری بین مدل چکیده توان پایین با مدل مشخصه فاقد توان است پرداخته شد. نتایج تجربی نشان دادند که متدولوژی پیشنهادی قادر است در مدت زمان قابل قبولی به درستی‌سنجی پردازنده‌هایی با هزاران خط کد UCLID و ده‌ها حوزه‌ی توانی بپردازد. از جمله کارهای آینده‌ای که مدنظر داریم، عیب‌یابی و تصحیح ایرادهای طراحی است. در صورتیکه مدل چکیده با مشخصه یکسان نباشد یا ویژگی‌های غیرعملکردی برقرار نباشند، چطور می‌توان مشخص کرد که کدام بخش از طراحی منجر به این ایراد شده است.

مراجع

[1] IEEE Standard for Design and Verification of Low-Power, Energy-Aware Electronic Systems, IEEE Standard 1801-2015, UPF-3.0, Unified Power Format 3.0, 2016.

[2] O. Mbarek, A. Pegatoquet, and M. Auguin, "Using unified power format standard concepts for power-aware design and verification of systems-on-chip at transaction level," IET Circuits, Devices & Systems, vol. 6, no. 5, pp. 287-296, 2012.

[3] M. Siro, D. Macko, and K. Jelemenská, "PMS2UPF: An automated transition from ESL to RTL power-intent specification," 20th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS), pp. 140-144, 2017.

[4] A. Hazra, S. Goyal, P. Dasgupta, and A. Pal, "Formal verification of architectural power intent," IEEE Transactions on Very Large Scale Integration (TVLSI) Systems, vol. 21, no. 1, pp. 78-91, 2013.

[5] A. Hazra, R. Mukherjee, P. Dasgupta, A. Pal, K. M. Harer, A. Banerjee, et al., "POWER-TRUCTOR: An integrated tool flow for formal verification and coverage of architectural power intent," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), vol. 32, no. 11, pp. 1801-1813, 2013.

[6] C. C.-H. Hsu and C. H.-P. Wen, "Speeding up power verification by merging equivalent power domains in RTL design with UPF," International,Test Conference in Asia (ITC-Asia), pp. 168-173, 2017.

[7] C.-P. Lu and I. H.-R. Jiang, "Fast low power rule checking for multiple power domain design," in 2017 Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 1745-1750, 2017.

[8] C. Eisner, A. Nahir, and K. Yorav, "Functional verification of power gated designs by compositional reasoning," Formal Methods in System Design, vol. 35, no.1, pp. 40-55, 2009.

[9] A. M. Gharehbaghi and M. Fujita, "Specification and formal verification of power gating in processors," 15th International Symposium on Quality Electronic Design (ISQED), pp. 604-610, 2014.

[10] R. Sharafinejad, B. Alizadeh, and M. Fujita, "UPF-based formal verification of low power techniques in modern processors," 33rd IEEE VLSI Test Symposium (VTS), pp. 1-6, 2015.

[11] S. Mandal, A. B. Da Costa, A. Hazra, P. Dasgupta, B. Naware, R. M. Chunduri, et al., "Formal Verification of Power Management Logic with Mixed-Signal Domains," 30th International Conference on VLSI Systems Design (VLSID) , pp. 239-244, 2017.

[12] D. Macko, K. Jelemenská, and P. Čičák, "Verification of Power-Management Specification at Early Stages of Power-Constrained Systems Design," Journal of Circuits, Systems and Computers, vol. 26, no. 8, pp. 1-23, 2017.

[13] R. Sharafinejad, B. Alizadeh, and Z. Navabi, "Automatic Correction of Dynamic Power Management Architecture in Modern Processors," IEEE Transactions on Very Large Scale Integration (TVLSI) Systems, vol. 26, no. 2, pp. 308-318, 2018.

[14] JasperGold Low-Power Verification App. [Online]. Available: https://www.cadence.com/jasper-gold-verificationplatform/low-power-verification-app.html

[15] Synopsys Low Power Verification Tools Suite User Guide. [Online]. Available: http//solvent.synopsys.com

[16] B. A. Brady, S. A. Seshia, S. K. Lahiri, and R. E. Bryant, "A User's Guide to UCLID Version 3.1," 2008.

[17] J. R. Burch and D. L. Dill, "Automatic verification of pipelined microprocessor control," in International Conference on Computer Aided Verification, 1994, pp. 68-80.

[18] MIPS32 architecture. [Online]. Available: http://www.imgtec.com/mips/mips32-architecture.asp

[19] J. Gaisler, E. Catovic, M. Isomaki, K. Glembo, and S. Habinc, "GRLIB IP core user's manual," Gaisler research, 2007.

[20] K. Marcinek and W. A. Pleskacz, "ELEON3LP–Superscalar and low-power enhancements of single issue general purpose processor model," Microprocessors and Microsystems (MICPRO), vol. 37, no. 6, pp. 693-700, 2013.

زیرنویس‌ها

[1]Power management strategy
[2]System level
[3]Low-power intent
[4]Unified Power Format (UPF)
[5]Register-transfer level (RTL)
[6]Gate level
[7]Transaction level (TL)
[8]Electronic system level (ESL)
[9]Power domain
[10]cross-domain path
[11]Specification model
[12]Power Management Unit (PMU)
[13]Isolation cell
[14]Retention Register (RR)
[15]Local Power Management (LPM)
[16]Power mode
[17]Global Power Management (GPM)
[18]Legal power state
[19]Low-power implementation model
[20]Extraction of power feature
[21]State machine
[22]Abstracted low-power model
[23]Data-path component
[24]Always-on
[25] Retaining value itself
[26] Losing value
[27] Retaining value in external memory
[28] Uninterpreted function
[29] Interpreted function
[30] Predicate fucntion
[31] Instruction Set Architecture (ISA)
[32] Pipelined implementation model
[33] Flush
[34]Symbolic simulation
[35]Correspondence checking
[36]Power State Table (PST)
[37]Always-on power domains
[38]Conflict power domains
[39]Coherent power domain
[40]Illegal power state