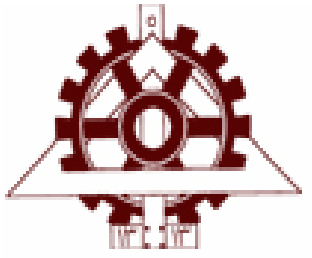
به نام خدا



دانشگاه تهران

دانشکدهٔ فنی

گروه مهندسی برق و کامپیوتر

عنوان:

ساختمان‌داده بهینه برای نمایش سطح انتقال‌ثبات

نگارش:

پژمان لطفی‌کامران

استاد راهنما:

آقای دکتر زین‌العابدین نوابی‌شیرازی

استاد مشاور:

آقای دکتر مهران معصومی

پایان‌نامه برای دریافت درجهٔ کارشناسی ارشد در رشتهٔ

مهندسی کامپیوتر- گرایش معماری کامپیوتر

اسفند ماه ۱۳۸۳

به نام خداوند جان و خرد

کزین برتر اندیشه بر نگذرد

دانشگاه تهران

دانشکدهٔ فنی

گروه مهندسی برق و کامپیوتر

عنوان:

ساختمان‌داده بهینه برای نمایش سطح انتقال‌ثبات

نگارش:

پژمان لطفی‌کامران

پایان‌نامه برای دریافت درجهٔ کارشناسی ارشد در رشتهٔ
مهندسی کامپیوتر- گرایش معماری کامپیوتر

از این پایان‌نامه در تاریخ ۱۳۸۳/۱۲/۹ در مقابل هیات داوران دفاع به عمل آمد و مورد تصویب قرار گرفت.

معاونت تحصیلات تکمیلی دانشکده فنی:	**دکتر جواد فیض**
مدیر گروه آموزشی:	**دکتر پرویز جبه‌دار مارالانی**
سرپرست تحصیلات تکمیلی گروه:	**دکتر سید مهدی فخرایی**
استاد راهنما:	**دکتر زین‌العابدین نوابی‌شیرازی**
استاد مشاور:	**دکتر مهران معصومی**
عضو هیات داوران:	**دکتر امید فاطمی**
عضو هیات داوران:	**دکتر علی افضلی‌کوشا**
عضو هیات داوران:	**دکتر شاهین حسابی**

تقدیم به پدر و مادر عزیزم

تقدیر و تشکر

در ابتدا خاضعانه به درگاه خداوند متعال بخاطر الطاف بی‌پایانش شکرگزاری می‌نمایم.
از خانواده گرامی‌ام و بویژه پدر و مادر عزیزم که لحظه لحظه زندگیم با مهر محبت بی‌دریغشان رونق گرفته است، تشکر می‌نمایم.
از استاد عزیزم جناب آقای دکتر زین‌العابدین نوابی بخاطر زحمات و راهنمایی‌های ارزشمندشان سپاسگذارم. همچنین از استاد مشاورم جناب آقای دکتر مهران معصومی سپاسگذاری می‌نمایم. از اساتید ارجمند آقایان دکتر امید فاطمی، دکتر علی افضلی‌کوشا و دکتر شاهین حسابی که داوری این پایان‌نامه را به عهده گرفتند، تشکر می‌نمایم.
بر خود لازم می‌دانم از دوست و استاد عزیزم آقای محمد حسین‌آبادی که در طی این مدت کمک‌های فراوان و راهنمایی‌های ارزشمندشان همواره راهگشای من بوده است، نهایت سپاسگذاری را به عمل آورم. از دوستان عزیزم در گروه تحقیقاتی آزمایشگاه سخت‌افزار مخصوصا اعضای گروه ارزیابی‌رسمی آقایان شجاعی، افشار، کاکوئی و نادری که صمیمانه من را در انجام این پایان‌نامه یاری داده‌اند، تشکر می‌نمایم. همچنین از دوستان عزیزم آقایان هوشمند، شمشیری، علی‌صفایی و علیزاده که مرا در تهیه مطالب فصل چهارم این پایان‌نامه یاری داده‌اند، تشکر و قدرشناسی می‌نمایم. لازم می‌دانم از دوستان عزیزی که زحمت بازبینی این پایان‌نامه را به عهده گرفتند؛ آقایان اسمعیل‌زاده، کوخازاده، صحفی و شهابی تشکر و قدردانی نمایم. از سرکار خانم عسگری نیز بخاطر همکاری صمیمانه‌شان سپاسگذاری می‌نمایم.
در پایان از همه دوستان و عزیزانی که از کمک‌هایشان در مراحل مختلف انجام این پایان‌نامه استفاده نموده‌ام، کمال تشکر و قدردانی را خواهم داشت.

چکیده

امروزه شرکت‌های پیشرو در زمینه طراحی تراشه‌ها و سیستم‌های دیجیتال، بیش از هر زمان دیگری با مساله ارزیابی طراحی‌های انجام شده روبرو هستند. نتیجه مطالعات صنعتی آشکار می‌سازد که تا ۵۰٪ کل زمان تولید تراشه صرف ارزیابی آن می‌گردد. حتی شرکت‌های بزرگ با منابع نامحدود و با هزینه کردن افراد و زمان شبیه‌سازی بسیار نمی‌توانند در مدت زمان کوتاه و یا قابل پیش‌بینی به طرح بدون اشکال و نقص دست پیدا کنند.

پیچیدگی طرح‌های امروزی به نقطه‌ای رسیده است که ارزیابی مهمترین گلوگاه آن است. طراح تراشه می‌تواند تشخیص دهد که چه موقع کارش را به پایان رسانیده است؛ اما نمی‌تواند مطمئن شود که ارزیابی طرح به درستی انجام گرفته است چرا که روش‌های ارزیابی امروزی تجربی هستند. به ناچار باید انواع شبیه‌سازی‌ها را ادامه داد تا زمانی که تصمیم گرفته شود که آنچه انجام شده کافی می‌باشد. این امر بدلیل آن نیست که کاملا مطمئن گشته‌ایم که طرح خالی از اشکال است؛ بلکه بدلیل آن است که زمان به انتها رسیده است.

ارزیابی‌رسمی یک روش مطمئن برای برطرف کردن اشکالات و تکمیل روش شبیه‌سازی می‌باشد. ارزیابی‌رسمی اشکالات شبیه‌سازی را با اعمال قواعد ریاضی برای اثبات صحت پیاده‌سازی، برطرف می‌نماید.

با افزایش پیچیدگی و اندازه سیستم‌های دیجیتال لازم است که ارزیابی‌رسمی طرح‌ها در مراحل آغازین چرخه طراحی صورت گیرد. این امر نیازمند ابزارهای ارزیابی خودکار در سطوح بالا (سطح انتقال‌ثبات یا سطح رفتاری) می‌باشد. هم اکنون ابزارهایی برای ارزیابی‌رسمی طرح‌ها در سطوح انتزاع پایین مثل سطح گیت وجود دارد؛ اما ابزارهایی که بتوانند توصیفات سطح بالا را ارزیابی‌رسمی کنند، هنوز به سطح قابل قبولی نرسیده‌اند. این مساله تا قسمتی از نبود یک نمایش مناسب برای طرح‌های انتقال‌ثبات ناشی شده است. از آنجا که دیاگرام تصمیم‌گیری دودویی و مشتقاتش طرح‌های در سطح گیت را به نحو موثری نمایش می‌دهند، بسیاری از ابزارهای ارزیابی، توصیفات سطح انتقال‌ثبات را سنتز می‌کنند تا بتوانند از مزایای دیاگرام تصمیم‌گیری دودویی که در سطح گیت می‌باشد، بهره ببرند.

هدف از این پایان‌نامه فراهم آوردن بخشی از زیرساخت‌های لازم برای ارزیابی‌رسمی مدارهای دیجیتال در سطح انتقال‌ثبات می‌باشد. یکی از مهمترین لازمه‌های ارزیابی‌رسمی در سطح انتقال‌ثبات، توانایی نمایش یگانی توصیفات انتقال‌ثبات بوسیله یک ساختمان‌داده بهینه است. این ساختمان‌داده می‌تواند همان موفقیت‌هایی را که دیاگرام تصمیم‌گیری دودویی برای ارزیابی‌رسمی در سطح گیت به ارمغان آورد، برای ارزیابی‌رسمی در سطح انتقال‌ثبات به ارمغان آورد. بعد از بررسی ساختمان‌داده‌های موجود، دیاگرام بسط تیلور به عنوان زیرساخت ساختمان‌داده جدید برگزیده شد. در این پایان‌نامه سعی خواهد شد که با اعمال تغییرات و اضافه‌کردن قابلیت‌هایی به دیاگرام بسط تیلور، ساختمان‌داده مناسبی برای سطح انتقال‌ثبات پیشنهاد داده شود.

فهرست مطالب

فهرست اشکال

فهرست جداول

فهرست روابط

فصل اول: پیش‌گفتار

امروزه شرکت‌های پیشرو در زمینه طراحی تراشه‌ها و سیستم‌های دیجیتال، بیش از هر زمان دیگری با مساله ارزیابی[1] طراحی‌های انجام شده روبرو هستند. نتیجه مطالعات صنعتی آشکار می‌سازد که تا ۵۰٪ کل زمان تولید تراشه صرف ارزیابی آن می‌گردد [1].

پیچیدگی طرح‌های امروزی به درجه‌ای رسیده است که ارزیابی مهمترین گلوگاه[2] آن است [1]. طراح تراشه می‌تواند تشخیص دهد که چه موقع کارش را به پایان رسانیده است؛ اما نمی‌تواند مطمئن شود که چه موقع ارزیابی طرح به پایان رسیده است چرا که روش‌های ارزیابی امروزی تجربی[3] هستند. به ناچار باید انواع شبیه‌سازی‌ها [2] را ادامه داد تا زمانی که تصمیم گرفته شود که آنچه انجام شده کافی می‌باشد. این امر بدلیل آن نیست که کاملا مطمئن گشته‌ایم که طرح خالی از اشکال است؛ بلکه بدلیل آن است که زمان به انتها رسیده است.

در شبیه‌سازی[4] [2] سعی می‌شود که همه ورودی‌های ممکن به طرح اعمال شده و خروجی‌ها به دقت بررسی شوند. از تجربیات گذشته می‌دانیم که همه اشکالات با این شیوه پیدا نمی‌شوند. بنابراین سعی می‌شود که با افراد و زمان شبیه‌سازی بیشتر، تعداد بیشتری از اشکالات را کشف نمائیم. حتی شرکت‌های بزرگ با منابع نامحدود و با هزینه کردن افراد و زمان شبیه‌سازی بسیار نمی‌توانند در مدت زمان کوتاه و یا قابل پیش‌بینی به طرح بدون اشکال و نقص دست پیدا کنند [1].

ارزیابی‌رسمی[5] [2][3] یکی از راهکارهایی است که برای برطرف کردن اشکالات شبیه‌سازی [2] پیشنهاد داده شده است. در ارزیابی‌رسمی از روش‌های مبتنی بر ریاضی برای بررسی صحت طراحی استفاده می‌شود. برای آنکه تفاوت شبیه‌سازی و ارزیابی‌رسمی مشخص گردد، بررسی برابری[6] دو مدار سطح گیت را در نظر بگیرید. برای آنکه بوسیله شبیه‌سازی برابری این دو مدار بررسی گردد، باید همه ورودی‌های ممکن به دو مدار اعمال گردد و خروجی‌های این دو مدار در همه حالت‌ها با هم مقایسه شوند. اگر برای همه ورودی‌های ممکن، خروجی‌های دو مدار با هم برابر باشند، این دو مدار معادل می‌باشند و در غیر این صورت عدم برابری این دو به اثبات می‌رسد. اما روش ارزیابی‌رسمی برای بررسی برابری به شیوه دیگری عمل می‌کند. مدارهای دیجیتال از قرار گرفتن گیت‌های بولی کنار یکدیگر بوجود آمده‌اند. در روش ارزیابی‌رسمی سعی می‌شود که با استفاده از

[1] Verification
[2] Bottleneck
[3] Empirical
[4] Simulation
[5] Formal Verification
[6] Equivalence Checking

روش‌های جبر بول[1] ساده‌ترین عبارت بولی توصیف‌کننده هر کدام از خروجی‌های دو مدار بدست آید. با مقایسه ساده‌ترین عبارات بولی خروجی‌های متناظر دو مدار، می‌توان برابری آن دو را تحقیق کرد. همانطور که مشخص است در روش ارزیابی‌رسمی نیازی به اعمال ورودی به مدار نیست، بلکه برای ارزیابی از روش‌های مبتنی بر ریاضی استفاده می‌شود. روش‌های ارزیابی‌رسمی به دو دسته مهم تقسیم می‌شوند: بررسی برابری[2] [4][5] و بررسی خصوصیت[3] [6][7].

۱-۱ بررسی برابری

برای بررسی برابری مدارهای دیجیتال، همانطور که پیش از این عنوان گردید، ابتدا سعی می‌شود که ساده‌ترین توصیف بولی خروجی‌های دو مدار به دست آید. با مقایسه این دو توصیف بولی می‌توان برابری یا عدم برابری دو مدار را نتیجه گرفت. این روش با عبارات بولی سر و کار دارد، بنابراین باید بتواند به کمک ساختمان‌داده‌های[4] بهینه‌ای، عبارات بولی را در حافظه رایانه[5] نگهداری نماید. این ساختمان‌های‌داده دارای این ویژگی نیز می‌باشند که عبارات بولی را به ساده‌ترین شکل ممکن نمایش می‌دهند. در حقیقت با تبدیل یک عبارت بولی به این ساختمان‌های‌داده، آن عبارت را به ساده‌ترین شکل ممکن در آورده‌ایم. از آنچه گفته شد می‌توان نتیجه گرفت که کار اصلی در بررسی برابری دو مدار به عهده این ساختمان‌های‌داده است. تنها کاری که ما انجام می‌دهیم آن است که عبارات بولی توصیف کننده مدارها را به این ساختمان‌داده‌ها تبدیل می‌کنیم و ساختمان‌های‌داده حاصل را با هم مقایسه می‌نمائیم.

۱-۲ بررسی خصوصیت

در این روش بررسی می‌گردد که آیا یک خصوصیت یا ویژگی خاص در طرح وجود دارد یا نه. برای این منظور طرح به یک ماشین‌حالت[6] تبدیل می‌گردد. الگوریتم‌های بهینه‌ای وجود دارد که به یک ماشین‌حالت اعمال می‌گردد و مشخص می‌سازد که آیا این ماشین‌حالت (یا به عبارت دیگر خود طرح) خصوصیت مورد نظر را دارا می‌باشد یا خیر. خیلی اوقات حین اجرای این الگوریتم‌ها لازم می‌شود که برابری یا عدم برابری دو تابع بررسی گردد [6][7]. برای این منظور رابطه بین حالت‌ها، توابع محلی هر حالت و توابع خروجی طرح به

[1] Boolean Algebra
[2] Equivalence Checking
[3] Property Checking
[4] Data Structure
[5] Computer
[6] State Machine

ساختمان‌داده‌هایی که در بررسی برابری استفاده می‌شوند، تبدیل می‌گردند. در نتیجه، هر زمان که لازم باشد برابری دو بخش بررسی گردد، صرفا ساختمان‌داده‌های متناظر آن دو بخش با هم مقایسه می‌گردند.

۱-۳ ساختمان‌های‌داده ارزیابی‌رسمی

همانطور که در بخش‌های قبلی دیدیم، ساختمان‌های‌داده که عبارات بولی را به ساده‌ترین شکل ممکن نمایش می‌دهند، از اهمیت ویژه‌ای در ارزیابی‌رسمی برخوردار می‌باشند. مهمترین و پرکاربردترین ساختمان‌داده که در ابزارهای ارزیابی‌رسمی استفاده می‌شود، دیاگرام تصمیم‌گیری دودویی[1] می‌باشد. این ساختمان‌داده عبارات بولی را به ساده‌ترین شکل ممکن و به صورت بسیار فشرده نمایش می‌دهد. این ساختمان‌داده برای نمایش طرح‌های سطح گیت مناسب می‌باشد؛ اما در نمایش توصیفات انتقال‌ثبات ضعیف عمل می‌کند.

علت عدم نمایش بهینه توصیفات انتقال‌ثبات بوسیله دیاگرام تصمیم‌گیری دودویی آن است که همه عملگرهای سطح انتقال‌ثبات (مثلا ضرب‌کننده‌ها) باید به معادل بولی خود تبدیل شوند؛ چرا که دیاگرام تصمیم‌گیری دودیی تنها قادر است عبارات بولی را نمایش دهد. معادل بولی عملگرهای سطح انتقال‌ثبات بزرگ و پیچیده می‌باشند و نگهداری آنها در حافظه به کمک دیاگرام تصمیم‌گیری دودویی بهینه نمی‌باشد.

برای حل این مشکل باید ساختمان‌داده مناسبی برای نمایش سطح انتقال‌ثبات ارائه شود. پس از آن دیگر لازم نخواهد بود که عملگرهای سطح انتقال‌ثبات را به معادل بولی‌شان تبدیل کرد. هدف از این پایان‌نامه ارائه یک ساختمان‌داده بهینه برای نمایش سطح انتقال‌ثبات می‌باشد. پس از بررسی ساختمان‌داده‌های موجود، دیاگرام بسط تیلور[2] به عنوان زیربنای ساختمان‌داده جدید برگزیده شد. دیاگرام بسط تیلور محدودیت‌هایی دارد که آنرا برای نمایش سطح انتقال‌ثبات مناسب نمی‌سازد. این محدودیت‌ها عبارتند از:

- پیچیدگی ساختمان‌داده دیاگرام بسط تیلور و عدم وجود یک پیاده‌سازی بهینه برای آن
- ضعف در نمایش عبارات بولی سطح بیت
- عدم توانایی در نمایش عبارات بولی سطح‌کلمه

در این پایان‌نامه سعی شده است که با برطرف کردن اشکالات دیاگرام بسط تیلور، به یک ساختمان‌داده بهینه برای نمایش سطح انتقال‌ثبات دست یافت. در نوشتن این پایان‌نامه سعی شده است که حتی‌المقدور فصل‌ها مستقل از هم باشند. برای این منظور در برخی موارد اطلاعات تکراری ارائه شده است. البته این اطلاعات کم

[1] Binary Decision Diagram
[2] Taylor Expansion Diagram

هستند و خواننده پس از آشنایی با مطالب به راحتی می‌تواند از روی موارد تکراری عبور کرده و به مبحث اصلی بپردازد.

این پایان‌نامه به صورت زیر سازماندهی شده است. در فصل دوم بر آنچه در زمینه ساختمان‌های‌داده ارزیابی‌رسمی در طی دو دهه گذشته انجام شده است، مروری مختصر صورت خواهد گرفت. در فصل سوم دیاگرام بسط تیلور مورد بحث و بررسی قرار خواهد گرفت. در فصل چهارم راه‌حلی بهینه برای پیاده‌سازی دیاگرام بسط تیلور ارائه می‌گردد، که به این ترتیب مشکل اول دیاگرام بسط تیلور برطرف می‌شود. در فصل پنجم راه‌حلی برای بهبود نمایش عبارات بولی سطح بیت ارائه می‌گردد. البته این فصل راه‌حل نهایی بهبود نمایش عبارات بولی سطح بیت را ارائه نمی‌کند. راه‌حل نهایی در فصل ششم ارائه می‌گردد. این فصل بیشتر از منظر ریاضیات موضوع را مورد بحث قرار می دهد؛ ولی کاربردی کردن ایده با مشکلاتی مواجه است. در فصل ششم راه‌حل نهایی برای بهبود نمایش عبارات بولی سطح بیت ارائه می‌گردد. به این ترتیب مشکل دوم دیاگرام بسط تیلور هم برطرف می‌گردد. فصل هفتم راه‌حلی برای برطرف کردن مشکل سوم دیاگرام بسط تیلور، نمایش عبارات بولی سطح‌کلمه، ارائه می‌دهد. نهایتا، دست‌آوردهای این پایان‌نامه و کارهای آینده در فصل هشتم ارائه می‌گردند.

فصل دوم: دیاگرام‌های تصمیم‌گیری

دیاگرام تصمیم‌گیری دودویی مرتب‌شده[1] در زمینه‌های متعددی از قبیل ارزیابی‌رسمی[2]، سنتز[3] و تولید تست[4] کاربرد دارد. این دیاگرام توابع بولی[5] را به صورت یگانی[6] و فشرده نمایش می‌دهد. همچنین این دیاگرام را می‌توان بوسیله الگوریتم‌های گراف به نحو موثری تولید کرد. محققان دریافته‌اند که بسیاری از کارها را می‌توان بوسیله یکسری عملیات روی توابع بولی مدلسازی کرد. این امر موجبات استفاده گسترده از دیاگرام تصمیم‌گیری دودویی مرتب‌شده را فراهم آورده است.

به دلیل این موفقیت‌ها، تلاش بسیاری برای بهبود عملکرد و گسترش حوزه عمل دیاگرام تصمیم‌گیری دودویی انجام گرفته است. تکنیک‌هایی بدست آمده که این نمایش را فشرده‌تر می‌سازد و بدین وسیله باعث می‌شوند که این دیاگرام مجموعه وسیعتری از توابع را در بر گیرد. این امر موجب کارایی بیشتر کاربردهای مبتنی بر دیاگرام تصمیم‌گیری دودویی و گسترش مجموعه مسائل قابل حل گردیده است.

در این فصل مروری خواهیم داشت بر نمایش‌های مبتنی بر گراف و به نمایش‌هایی که اهمیت بیشتری در ارزیابی‌رسمی داشته‌اند، دقیق‌تر خواهیم پرداخت.

۲-۱ مقدمه

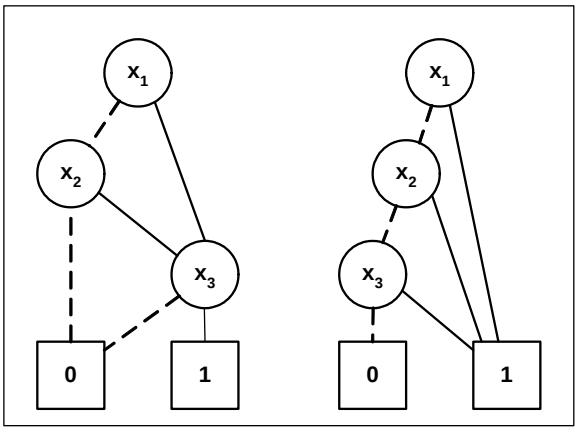
اگرچه ایده نمایش توابع بولی به صورت گراف‌های تصمیم‌گیری، ریشه در تاریخ دارد (برای نمونه مقاله اکرز[7] [8] را ببینید)، استفاده گسترده از آنها به عنوان ساختمان‌داده‌ای برای نمایش نمادین[8] توابع بولی، از سال ۱۹۸٦ و پس از ارائه یکسری الگوریتم برای ساختن و انجام عملیات مختلف روی این ساختمان‌داده، آغاز گردید [13]. یکی از نکات کلیدی در این الگوریتم‌ها، اعمال یکسری محدودیت‌ها روی ترتیب متغیرها می‌باشد، بدین معنا که ترتیب متغیرها در هر مسیر از ریشه به برگ یکسان می‌باشد.

از سال ۱۹۸٦، فعالیت روی دیاگرام تصمیم‌گیری دودویی گسترش یافت [15] و این دیاگرام کاربردهای فراوانی خصوصا در زمینه ابزارهای طراحی به کمک کامپیوتر[9] پیدا کرد. با اصلاحاتی در ساختمان‌داده و الگوریتم‌های آن، زمان انجام عملیات و حافظه مصرفی بهبود پیدا کرد. ایده اصلی دیاگرام تصمیم‌گیری دودویی برای نمایش

[1] Ordered Binary Decision Diagram
[2] Formal Verification
[3] Synthesis
[4] Test Generation
[5] Boolean Functions
[6] Canonical
[7] Akers
[8] Symbolic Representation
[9] Computer-Aided Design

موثرتر مجموعه‌های دیگری از توابع، گسترش پیدا کرد. تحقیقات گسترده در این زمینه منجر به یکسری از اسامی چندحرفی و شبیه هم شده است. آنچه در زیر آورده شده تنها بخشی از آنها می‌باشند: OBDD، FBDD، FDD، OKFDD، EVBDD، MTBDD، BMD، *BMD، ZBDD، ABDD، HDD، TDD و OPDD. دانستن آنکه این دیاگرام‌ها چگونه پیاده‌سازی می‌شوند و چگونه به هم مربوط می‌گردند، کار سختی است.

در این فصل، پیشرفت‌های هجده سال گذشته در زمینه نمایش‌های مبتنی بر گراف، ارائه می‌گردد. از آنجا که بررسی همه موارد عملی نیست، تنها آن دسته از روش‌ها که تاثیر قوی‌تری خصوصا در عرصه ارزیابی‌رسمی داشته‌اند، بررسی خواهند شد.



شکل ۲-۱: مثال‌هایی از دیاگرام تصمیم‌گیری دودویی

۲-۲ پیش زمینه

در دیاگرام تصمیم‌گیری دودویی، یک تابع به صورت یک گراف نمایش داده می‌شود و هر گره غیرترمینال بوسیله یکی از متغیرهای تابع[1] برچسب[2] زده می‌شود. در شکل ۲-۱ نمایش دیاگرام تصمیم‌گیری دودویی مربوط به توابع $(x_1 \vee x_2) \wedge x_3$ (چپ) و $x_1 \vee x_2 \vee x_3$ (راست) نشان داده شده است. از هر گره دو یال خارج می‌گردد، متناظر با مواقعی که متغیر گره به مقدار 0 (به صورت خط چین نشان داده شده است) یا به مقدار 1 (به صورت خط ممتد نشان داده شده است) نسبت داده شده است. گره‌های ترمینال (به صورت مربع نشان داده شده است) با 0 یا 1 برچسب زده شده‌اند و متناظر با خروجی‌های تابع می‌باشند. برای هر ورودی مشخص،

[1] Function Variables
[2] Label

خروجی تابع با پیمودن یک مسیر از ریشه به یک گره ترمینال و با انتخاب شاخه مناسب در هر گره، مشخص می‌گردد.

گراف‌های شکل ۲-۱ مثال‌هایی از دیاگرام تصمیم‌گیری دودویی مرتب‌شده می‌باشند، بدین معنا که اگر بین متغیرها ترتیب $x_1 < x_2 < x_3$ را داشته باشیم، آنگاه در هر مسیر از ریشه به برگ متغیرها به صورت صعودی ظاهر می‌شوند. با اعمال یکسری قواعد ساده‌ساز[1] [13] می‌توان یک دیاگرام تصمیم‌گیری دودویی مرتب‌شده را به فرم یگانی[2] (یک نمایش یکتا برای آن ترتیب از متغیرها) تبدیل کرد.

بسیاری از عملیات روی توابع بولی را می‌توان به صورت الگوریتم‌های ساده گراف که روی دیاگرام تصمیم‌گیری دودویی مرتب‌شده عمل می‌نمایند، پیاده‌سازی کرد [13][15]. برای مثال: تشخیص اینکه آیا دو تابع یکسان هستند، تولید تابع متناظر با AND، OR و NOT دیگر توابع و یا تشخیص تعداد ترکیب‌های ورودی که منجر به یک شدن خروجی می‌گردند. پیچیدگی زمانی و حافظه‌ای این الگوریتم‌ها به صورت چندجمله‌هایی از اندازه گراف‌های ورودی‌شان (عملوندهای این عملگرها) می‌باشد. آنها همچنین دارای این ویژگی می‌باشند که همه دیاگرام‌های تصمیم‌گیری دودویی که از یک ترتیب واحد بین متغیرها تبعیت می‌کنند، یگانی بوده و این ویژگی، آنها را برای عملیات بعدی مناسب می‌سازد.

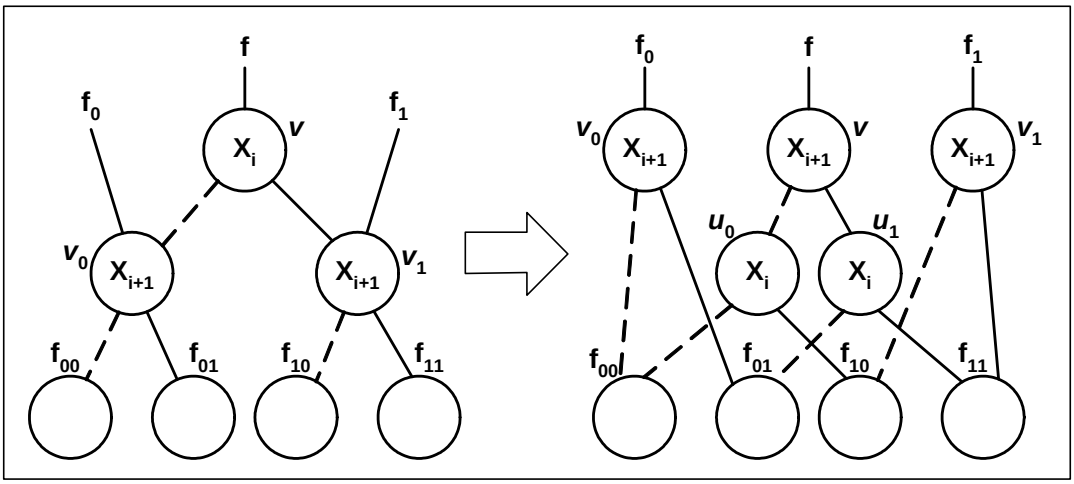
برای اعمال دیاگرام تصمیم‌گیری دودویی به یک مساله، ابتدا باید داده‌ها را به توابع بولی تبدیل کرد. کارها به صورت یکسری از مراحل که هر کدام یک عمل روی دیاگرام تصمیم‌گیری دودویی می‌باشد، ادامه می‌یابد. برای مثال عمل ساختن دیاگرام تصمیم‌گیری دودویی برای تابع خروجی یک مدار ترکیبی را در نظر بگیرید. متغیرهایی برای ورودی‌های اصلی تعریف و دیاگرام تصمیم‌گیری دودویی آنها ساخته می‌شود. سپس مدار به صورت نمادین ارزیابی شده و دیاگرام تصمیم‌گیری دودویی خروجی هر گیت با اعمال عملیات آن گیت روی دیاگرام تصمیم‌گیری دودویی ورودی‌های آن گیت، ساخته می‌شود. این عملیات برحسب توپولوژی[3] مدار ادامه می‌یابد تا دیاگرام تصمیم‌گیری دودویی همه خروجی‌های اصلی ساخته شود.

۲-۳ جابجایی متغیرها به صورت پویا

هنگام استفاده از دیاگرام تصمیم‌گیری دودویی، کاربر[4] باید یک ترتیب کلی برای متغیرهای تابع انتخاب نماید. در بسیاری از کاربردها[1]، یک ترتیب واحد برای همه توابع استفاده می‌شود. این امر مقایسه و ترکیب توابع

[1] Reduction Rules
[2] Canonical
[3] Topology
[4] User

مختلف را ممکن می‌سازد. کاربران دریافته‌اند که یافتن یک ترتیب مناسب میان متغیرها حیاتی است-- یک ترتیب مناسب میان متغیرها منجر به نمایش فشرده با مصرف حافظه قابل قبول می‌گردد؛ حال آنکه یک انتخاب ضعیف باعث اتمام حافظه حتی بزرگترین کامپیوترها می‌گردد. علاوه بر این، کارایی به علت استفاده از حافظه مجازی بسیار پایین می‌آید.



شکل ۲-۲: تغییر ترتیب متغیرهای همسایه در دیاگرام تصمیم‌گیری دودویی

اگرچه مقاله سال ۱۹۸٦ آقای برایانت[۲] [13] چندین قانون سرانگشتی[۳] برای انتخاب دستی یک ترتیب مناسب میان متغیرها بیان نمود، تنها پس از پیاده‌سازی چندین روش تجربی[٤] برای انتخاب خودکار[٥] یک ترتیب مناسب میان متغیرها (برحسب کاربرد) بود که استفاده از دیاگرام تصمیم‌گیری دودویی مرتب‌شده گسترده گردید. برای مثال، با داشتن یک مدار ترکیبی در سطح گیت، روش‌های تجربی می‌توانند یک ترتیب مناسب میان متغیرهای ورودی‌های اصلی مدار بیابند به نحوی که نمایش دیاگرام تصمیم‌گیری دودویی مرتب‌شده خروجی‌های اصلی تابع فشرده باشند [19][24]. کاربردهایی از قبیل مدارهای ترتیبی، کمتر از این روش‌ها سود می‌برند. به علاوه، در این روش‌ها یک ترتیب ثابت میان متغیرها بدست آمده و از ابتدا تا انتها از آن استفاده می‌شود. در عمل،

[1] Applications
[2] Bryant
[3] Rule of Thumb
[4] Heuristic
[5] Automatic

ترتیب ایده‌آل متغیرها با عبور برنامه از یک مرحله به مرحله بعد، ممکن است تغییر کند. نهایتا، هر کاربرد جدیدی به یک روش تجربی جدید برای انتخاب ترتیب مناسب میان متغیرها نیاز دارد.

ایده تغییر ترتیب متغیرها به صورت پویا[1] [31] که توسط رودل[2] ارائه گردید، بسیاری از این محدودیت‌ها را برطرف کرد. همانطور که برنامه پیش می‌رود، یک دیاگرام تصمیم‌گیری دودویی چندریشه‌ای بدست می‌آید که همه توابع مورد نظر را با یک ترتیب واحد میان متغیرها، نمایش می‌دهد. برنامه دائما تلاش می‌کند که ترتیب متغیرها را عوض نماید تا حافظه مصرفی را حداقل کند. این تغییرات می‌تواند در پشت صحنه و بدون دخالت کاربر انجام گیرد. بنابراین، جابجایی پویا با تغییر توابع انطباق پیدا کرده و برای همه کاربردها مناسب است.

یکی از دلائل مهم موفقیت جابجایی پویا، سهولت اضافه کردن آن به بسته‌های دیاگرام تصمیم‌گیری دودویی موجود می‌باشد. برای نمونه، در بسته‌ای که در [12] معرفی گردیده است، توابع با اشاره‌گرهایی به گره‌ها در فضای دیاگرام تصمیم‌گیری دودویی مشخص می‌گردند. هر گره یک شمارنده[3] دارد که نشان می‌دهد چه موقع هیچ اشاره‌گری به آن گره وجود ندارد و بنابراین حافظه مصرفی گره می‌تواند آزاد شود. آزاد کردن گره‌های اضافی در پشت صحنه و به صورت متوالی انجام می‌گیرد. بنابراین در هر زمان می‌توان تعداد اشاره‌گرها به یک گره را بدست آورد؛ اما نمی‌توان فهمید که این اشاره‌گرها کجا استفاده شده‌اند.

رودل جابجایی متغیرها را همزمان با آزاد کردن گره‌های اضافی انجام داد. بدین معنا که برنامه به صورت متناوب تلاش می‌کرد تا حافظه مورد نیاز را به کمک جابجایی متغیرها در فضای دیاگرام تصمیم‌گیری دودویی و آزاد کردن گره‌های اضافی، کمتر نماید. این برنامه جابجایی متغیرها را به کمک تعدادی تعویض در مکان دو متغیر همسایه انجام می‌داد. این تعویض‌ها و آزاد کردن‌ها هیچ تغییری در اشاره‌گرهای خارجی ایجاد نمی‌کرد و بنابراین می‌توانست بدون هیچ تاثیری روی برنامه (بجز در مواردی کاهش کارایی) انجام گیرند.

شکل ۲-۲ روش تعویض متغیرهای x_i و x_{i+1} را در فضای دیاگرام تصمیم‌گیری دودویی نشان می‌دهد. فرض کنید که در ترتیب اولیه (چپ)، تابع f بوسیله یک اشاره گر به گره v که با متغیر x_i برچسب خورده است، در فضای دیاگرام تصمیم‌گیری دودویی مشخص شده است. معمولا گره v با شاخه‌هایی به گره‌های v_0 و v_1 که هر دو با متغیر x_{i+1} برچسب خورده‌اند و خود نیز با شاخه‌هایی به زیر گراف‌های f_{00} ، f_{01} ، f_{10} ، f_{11} متصل شده‌اند، متصل می‌گردد. پس از تعویض متغیرها، تابع f باید بوسیله یک اشاره گر به یک گره که با متغیر x_{i+1} برچسب خورده است، مشخص گردد. این گره باید با شاخه‌هایی به گره‌هایی که با متغیر x_i برچسب

[1] Dynamic Variable Reordering
[2] Rudell
[3] Counter

خورده‌اند و این گره‌ها نیز باید با شاخه‌هایی به زیر گراف‌های f_{00} ، f_{01} ، f_{10} ، f_{11} متصل گردند. این نتیجه در سمت راست نشان داده شده است. کماکان تابع f بوسیله یک اشاره گر به گره v نشان داده می‌شود و تمام اشاره‌گرها به توابع موجود (f_0 و f_1) دست نخورده باقی می‌مانند. در عوض گره‌های u_0 و u_1 اضافه شده‌اند. بنابراین تعویض متغیرها می‌تواند بدون تغییر اشاره‌گرهای خارجی در فضای دیاگرام تصمیم‌گیری دودویی صورت گیرد. اگرچه این شکل یک افزایش در اندازه گراف را نشان می‌دهد، در عمل گره‌های v_0 و v_1 ممکن است آزاد گردند و گره‌های u_0 و u_1 ممکن است قبلا ساخته شده باشند. اگر گره v به گره‌هایی که با متغیری بجز x_{i+1} برچسب خورده‌اند، متصل شده باشد، آنگاه تبدیلات مختلفی مورد نیاز است.

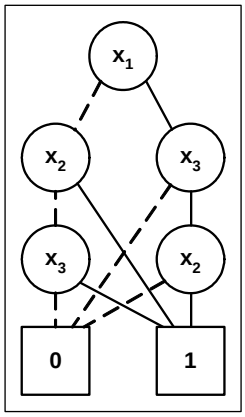
جابجایی متغیرها به صورت پویا خصوصا برای ارزیابی و تحلیل مدارهای ترتیبی مناسب می‌باشند (برای مثال نشان دادن اینکه دو مدار ترتیبی معادل می‌باشند). معمولا این کاربردها در چندین مرحله انجام می‌گیرند، اول ساختن نمایش تابع حالت بعدی[1]، احتمالا تبدیل کردن آن به یک رابطه گذر[2] و نهایتا انجام یک عملیات تکراری برای محاسبه مجموعه حالت‌های قابل دسترس[3]. اگرچه روش‌های تجربی مبتنی بر پیمودن شبکه گیت‌ها می‌توانند یک ترتیب مناسب برای نمایش تابع حالت بعدی بدست آوردند، این ترتیب اغلب برای مراحل بعدی مناسب نیست. همانطور که محاسبات انجام می‌گیرند و توابع تغییر می‌کنند، جابجایی پویا به نمایش اجازه می‌دهد که خود را با این تغییرات منطبق سازد. جابجایی پویا می‌تواند برنامه‌ها را تا ۱۰ برابر کند سازد، اما کاربران دریافته‌اند که این مساله می‌تواند اختلاف بین موفقیت و شکست در اتمام یک برنامه باشد.

۲-٤ کمرنگ کردن ترتیب متغیرها

اگرچه انتخاب یک ترتیب اولیه مناسب میان متغیرها و تغییر آن به صورت پویا همانطور که برنامه پیش می‌رود، می‌تواند حافظه مصرفی برنامه را کمتر نماید، مواردی وجود دارد که هیچ ترتیب مناسبی برای متغیرها وجود ندارد. برای مثال، دسته‌ای از توابع وجود دارند که اندازه دیاگرام تصمیم‌گیری دودویی مرتب‌شده آنها بدون توجه به ترتیب انتخاب شده، تابعی نمایی برحسب تعداد متغیرهای آن توابع می‌باشد [14]. این انفجار نمایی، اندازه مسائل قابل حل را محدود می‌نماید. توابعی که خروجی‌های یک ضرب کننده ترکیبی را نمایش می‌دهند، مثالی از یک دسته از این توابع می‌باشند. اچی[4] [27] توانست دیاگرام تصمیم‌گیری دودویی مرتب‌شده یک ضرب کننده ۱۵-بیتی را با موفقیت بسازد. برای انجام این کار بیش از ۱۲ میلیون گره (نزدیک ۰/۲٦ گیگابایت)

[1] Next State
[2] Transition Relation
[3] Reachable State
[4] Ochi

استفاده شد. با اضافه کردن یک بیت به اندازه کلمه، تعداد گره‌ها با ضریب تقریبی ۲/۷ افزایش می‌یابد و از این رو حتی قوی‌ترین کامپیوترها نیز نمی‌توانند از این حد عبور کنند.
یک استراتژی مشخص برای این کار آن است که محدودیت ترتیب میان متغیرها را حذف نموده یا کمرنگ[1] کنیم. برای مثال، اشار[2] همه محدودیت‌های نحوه قرار گرفتن متغیرها در گراف را حذف نمود [9]. یک متغیر می‌توانست در هر نقطه‌ای در طول یک مسیر و به هر تعداد که بخواهد، قرار گیرد. در این کرانه بیشتر خواص مورد نظر دیاگرام تصمیم‌گیری دودویی از بین خواهد رفت. گراف‌های ساخته شده با این شیوه یگانی نمی‌باشند و الگوریتم بسیاری از عملگرها، در بدترین حالت، پیچیدگی نمایی دارد.



شکل ۲-۳: مثالی از یک دیاگرام تصمیم‌گیری دودویی آزاد

یک روش محتاطانه این است که ترتیب قرار گرفتن متغیرها را تا اندازه‌ای کمرنگ کنیم که به فشرده‌سازی نمایش کمک کند اما خواص مورد نیاز دیاگرام تصمیم‌گیری دودویی مرتب‌شده را از میان نبرد. یک راه رسیدن به این هدف این است که اجازه دهیم متغیرها در هر ترتیبی اما حداکثر یکبار، در هر مسیر از ریشه به برگ قرار گیرند. به این ساختمان‌داده، دیاگرام تصمیم‌گیری دودویی آزاد گفته می‌شود. مثالی از دیاگرام تصمیم‌گیری دودویی آزاد در شکل ۲-۳ آورده شده است. گرگو[3] و مینل[4] [20] همچنین سایلینگ[5] و وگنر[6] [30]

[1] Relax
[2] Ashar
[3] Gergov
[4] Meinel
[5] Sieling
[6] Wegener

الگوریتم‌های موثری بر پایه این نمایش ارائه کردند. در هر دو مورد، آنها همچنین این محدودیت را اضافه کردند که همه توابع بوسیله یک ترتیب مشخص ساخته شوند. این بدان معنا است که برای هر ورودی مشخص، مسیر انتخاب شده در همه گراف‌ها از یک ترتیب واحد میان متغیرها تبعیت نماید. تفاوت میان این روش و دیاگرام تصمیم‌گیری دودویی مرتب‌شده در این است که؛ در این روش ترتیب متغیرها برای یک ورودی ممکن است با ترتیب متغیرها برای یک ورودی دیگر، متفاوت باشد.

برن[1] و دیگران نشان داده‌اند که بسادگی می‌توان چنین دیاگرام‌های تصمیم‌گیری دودویی آزادی را به بسته‌های دیاگرام تصمیم‌گیری دودویی مرتب‌شده اضافه کرد [11]. در پیاده‌سازی آنها، به تغییر ترتیب متغیرها بر حسب ورودی به صورت یک تبدیل در فضای ورودی تابع نگاه شده است. بسیاری از عملیات مستقل از چنین تبدیلاتی هستند به شرط آنکه تبدیل همه توابع به صورت هماهنگ انجام گیرد. آنها نشان داده‌اند که برخی توابع که پیش از این خودسر[2] بودند، با این روش به سادگی قابل نمایش هستند. متاسفانه برای ضرب‌کننده‌ها فقط یک بهره متوسط قابل تصور است؛ نشان داده شده است که هر نمایش دیاگرام تصمیم‌گیری دودویی آزاد این توابع، به صورت نمایی بر حسب تعداد متغیرها گسترش پیدا می‌کند [28].

۲-۵ تغییر تابع تجزیه

یک روش دیگر برای رسیدن به نمایشی فشرده برای توابع بولی، تغییر مفهوم گره‌ها می‌باشد. دیاگرام تصمیم‌گیری دودویی بر اساس تجزیه‌ای[3] از توابع بولی که اصطلاحا بسط شانون[4] نامیده می‌شود (اما اولین بار توسط بول[5] ارائه گردید)، ساخته می‌شود. می‌توان تابع f را بر حسب متغیر x به صورت زیر تجزیه کرد:

$$f = \bar{x} \wedge f_{\bar{x}} \vee x \wedge f_x \qquad \text{رابطه ۲-۱}$$

در این رابطه تابع f_x کوفاکتور مثبت[6] تابع f بر حسب متغیر x می‌باشد (حاصل جایگزین کردن متغیر x با مقدار 1). به طور مشابه $f_{\bar{x}}$ کوفاکتور منفی[7] تابع f بر حسب متغیر x می‌باشد (حاصل جایگزین کردن متغیر x با مقدار 0). به ازای هر ورودی x، یکی از بخش‌های رابطه ۲-۱ صفر می‌گردد. به عبارت دیگر، این تجزیه، تابع را بر حسب مقدار متغیر x به دو بخش مجزا تقسیم می‌کند. هر گره به همراه نوادگانش یک تابع بولی مثل

[1] Bern
[2] Intractable
[3] Decomposition
[4] Shannon
[5] Boole
[6] Positive Cofactor
[7] Negative Cofactor

f را نمایش می‌دهند. برای گرهی که با x برچسب خورده است، یکی از یالهای خروجی به زیر گراف f_x و دیگری به $f_{\bar{x}}$ اشاره می‌کند. با دنبال کردن یک مسیر از ریشه به برگ، کوفاکتورهای متوالی آن تابع را بدست می‌آوریم تا به یک مقدار ثابت برسیم.

می‌توان توابع تجزیه دیگری، بر حسب عملگر XOR تعریف کرد:

$$f = f_{\bar{x}} \oplus x \wedge f_{\delta x} \qquad \textbf{رابطه ۲-۲}$$

$$= f_x \oplus \bar{x} \wedge f_{\delta x} \qquad \textbf{رابطه ۲-۳}$$

که $f_{\delta x}$ تفاضل بولی[1] تابع f برحسب متغیر x می‌باشد ($f_{\delta x} = f_x \oplus f_{\bar{x}}$). رابطه ۲-۲ اصطلاحا ***رید-مولر***[2] یا ***دویو منفی***[3] و رابطه ۲-۳ ***دویو مثبت***[4] نامیده می‌شود. این تجزیه‌ها، شبیه بسط تیلور[5] توابع پیوسته می‌باشند. این روابط، یک تابع را به صورت، مقدار تابع برای یک x مشخص (0 یا 1) به همراه چگونگی تغییرات تابع بر حسب تغییرات x ($f_{\delta x}$) توصیف می‌کنند. از آنجا که متغیر x تنها دو مقدار مختلف اختیار می‌کند، این دو بخش در سری کفایت می‌کند.

کبسچول[6] و دیگران نمایش گرافی توابع بولی بر اساس بسط ***رید-مولر*** را پیشنهاد کردند. آنها این نمایش را دیاگرام تصمیم‌گیری تابعی[7] [21] نامیدند. این نمایش شبیه دیاگرام تصمیم‌گیری دودویی مرتب‌شده می‌باشد، با این تفاوت که دو یال خارج‌شونده از هر گره، کوفاکتور منفی و تفاضل بولی تابع برحسب متغیر گره می‌باشند. به عنوان مثال، اگر یال ممتد خارج‌شونده از هر گره را به عنوان تفاضل بولی تعبیر کنیم، می‌توان گراف‌های شکل ۲-۱ را دیاگرام تصمیم‌گیری تابعی دانست. با این تعبیر، این گراف‌ها توابع $(x_1 \oplus x_2) \wedge x_3$ (چپ) و $x_1 \oplus x_2 \oplus x_3$ (راست) را نمایش می‌دهند.

دیاگرام تصمیم‌گیری دودویی مرتب‌شده و دیاگرام تصمیم‌گیری تابعی ویژگی‌های مشترک بسیاری دارند؛ نمایش آنها یگانی است و پیچیدگی بسیاری از عملگرها در آنها چندجمله‌ای می‌باشد. چندین اختلاف مهم نیز میان این دو وجود دارد. اول، قواعد ساده‌ساز مختلفی برای این دو وجود دارد. قواعد ساده‌ساز برای این دو دسته از

[1] Boolean Difference
[2] Reed-Muller
[3] Negative Davio
[4] Positive Davio
[5] Taylor Expansion
[6] Kebschull
[7] Functional Decision Diagram

گراف‌ها در شکل ۲-٤ نشان داده شده است. هر دو شرایطی را نشان می‌دهند که تحت آن شرایط می‌توان یک گره را حذف و همه اشاره‌گرهای ورودی آن را به فرزندش اشاره داد. در هر دو مورد، می‌خواهیم چنین تبدیلی وقتی اتفاق بیفتد که تابع مستقل از متغیر آن گره است. اگر تابع f مستقل از متغیر x باشد، آنگاه میان کوفاکتورهایش رابطه $f_x = f_{\bar{x}} = f$ برقرار خواهد بود و تفاضل بولی آنها $f_{\delta x} = 0$ خواهد گردید. بنابراین، این تبدیل تحت شرایط نشان داده شده در سمت چپ شکل ۲-٤ برای دیاگرام تصمیم‌گیری دودویی و تحت شرایط سمت راست شکل ۲-٤ برای دیاگرام تصمیم‌گیری تابعی اتفاق می‌افتد.

دوم، لزوما دنبال کردن یک مسیر از ریشه به برگ در دیاگرام تصمیم‌گیری تابعی مرتب‌شده، خروجی تابع برای یک ورودی مشخص را بدست نمی‌دهد. به طور خاص، برای متغیر گره x، باید هر دو زیر گراف پیموده شده و حاصل آن دو با هم XOR گردد. به عبارت دیگر، کوفاکتور مثبت تابع f از رابطه $fx = f_{\bar{x}} \oplus f_{\delta x}$ بدست می‌آید. بنابراین ارزیابی توابع با رتبه‌ای خطی برحسب تعداد گره‌های گراف صورت می‌گیرد.

برای دسته‌ای از توابع، دیاگرام تصمیم‌گیری تابعی به صورت نمایی فشرده‌تر از دیاگرام تصمیم‌گیری دودویی می‌باشد اما عکس این قضیه هم درست می‌باشد. برای بدست آوردن مزایای هر دو، دریچسلر[1] و دیگران یک فرم پیوندی[2] به نام دیاگرام تصمیم‌گیری تابعی کرانکر[3] پیشنهاد کردند [18]. در نمایش آنها، به هر متغیر یک تابع تجزیه نسبت داده می‌شود که می‌تواند هر کدام از رابطه ۲-۱، رابطه ۲-۲ و رابطه ۲-۳ باشد. همه توابع باید از یک ترتیب مشخص میان متغیرها تبعیت کنند. همه گره‌هایی که با یک متغیر برچسب خورده‌اند، باید از یک تابع تجزیه استفاده کنند. در پیاده‌سازی آنها، انتخاب تابع تجزیه همزمان با جابجا کردن متغیرها انجام می‌گیرد. بدین معنا که در زمانی که متغیرها جابجا می‌گردند، برنامه توابع تجزیه مختلف را روی گره‌ها امتحان کرده و تلاش می‌کند اندازه گراف را کمتر کند. نتایج آنها نشان می‌دهد که به طور متوسط، اندازه گراف‌ها به اندازه ۳۵٪ کاهش یافته است.

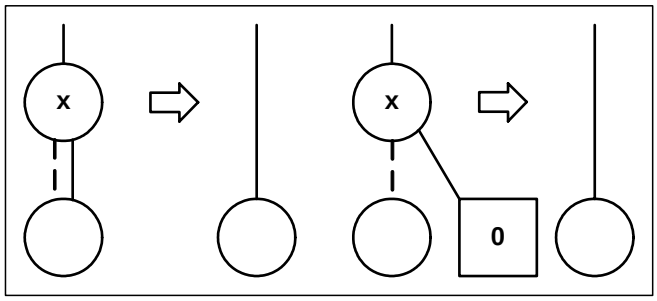
۲-٦ دیاگرام تصمیم‌گیری دودویی وقف‌شده به صفر

میناتو[4] نسخه‌ای از دیاگرام تصمیم‌گیری دودویی برای حل مسائل ترکیبی ارائه کرد. دیاگرام تصمیم‌گیری دودویی وقف‌شده به صفر[5] برای کاربردهایی که به نمایش مجموعه‌های خلوت[6] نیاز دارند، مناسبند [25]. فرض

[1] Drechsler
[2] Hybrid
[3] Ordered Kronecker Functional Decision Diagram
[4] Minato
[5] Zero Suppressed Binary Decision Diagram
[6] Sparse

کنید داده‌های یک مساله به صورت بردارهای بیتی[1] به طول n کد[2] شده است. هر زیر مجموعه از بردارها را می‌توان به صورت یک تابع بولی از n متغیر که خروجی آن تابع برای بردارهای موجود در آن مجموعه، یک می‌باشد، نمایش داد.

برای مثال، جمع حاصل‌ضرب‌های[3] توابع منطقی را در نظر بگیرید. این فرم‌ها معمولا به صورت یک مجموعه مکعب[4] نمایش داده می‌شوند که هر کدام از آنها بوسیله یک رشته شامل 0، 1 و – مشخص می‌گردند. برای مثال، تابع $(\bar{x}_1 \wedge x_2) \vee (\bar{x}_2 \oplus x_3)$ را می‌توان با مجموعه {01–,–11,–00} نمایش داد. برای کد کردن مکعب‌ها با بردارهای بیتی، هر علامت[5] را با دو بیت نمایش می‌دهیم. مثلا 10، 01 و 00 را برای نمایش علائم 1، 0 و – در نظر می‌گیریم. نهایتا مجموعه مکعب را با بردارهای بیتی {011000,001010,000101} نمایش می‌دهیم.



شکل ٢-٤: قواعد ساده‌ساز مختلف برای دیاگرام تصمیم‌گیری دودویی (چپ) و دیاگرام تصمیم‌گیری تابعی و دیاگرام تصمیم‌گیری دودویی وقف‌شده به صفر (راست)

در این مثال، مجموعه بردارهای بیتی از دو جهت خلوت می‌باشد. اول، تعداد بردارهای بیتی بسیار کمتر از 2^n بردار بیتی ممکن می‌باشد. دوم، خود بردارهای بیتی هم عناصر صفر زیادی دارند. در حقیقت کدگذاری علائم به نحوی صورت گرفته که چنین اتفاقی رخ دهد. این امر می‌طلبد که نمایشی به کار گرفته شود که از این دو ویژگی استفاده کند. دیاگرام تصمیم‌گیری دودویی وقف‌شده به صفر بسیار شبیه دیاگرام تصمیم‌گیری دودویی مرتب‌شده می‌باشد، بجز آنکه آنها از قواعد ساده‌ساز نشان داده شده در سمت راست شکل ٢-٤، بجای سمت چپ، استفاده می‌کنند. این بدان معنا است که یک گره زمانی حذف می‌گردد که با 1 قرار دادن متغیر آن گره،

[1] Bit Vectors
[2] Code
[3] Sum of Products
[4] Cube
[5] Symbol

تابع 0 شود. هنگامی که مجموعه‌ای از بردارهای بیتی را نمایش می‌دهیم، این شرط هنگامی اتفاق می‌افتد که داشتن 1 در یک موقعیت بیتی نشان دهد که آن بردار در آن مجموعه قرار ندارد. این شرط برای مجموعه‌های خلوت به کرات اتفاق می‌افتد و بنابراین گره‌های زیادی آزاد می‌گردند.

به عنوان مثال، هر دو گراف شکل ۲-۱ را می‌توان به صورت دیاگرام تصمیم‌گیری دودویی وقف‌شده به صفر تعبیر کرد، گراف سمت چپ مجموعه {101,011} و سمت راست مجموعه {100,010,001} را نمایش می‌دهد. در دیاگرام تصمیم‌گیری دودویی وقف‌شده به صفر توابع هر زیر گراف، به زمینه وابسته‌اند. برای مثال، در گراف سمت راست شکل ۲-۱ سه یال به ترمینال ۱ وجود دارد. هر کدام از این یال‌ها مفهوم متفاوتی دارند. بالاترین آنها مربوط است به وقتی که $x_2 = x_3 = 0$، وسطی به وقتی که $x_3 = 0$ و پایینی به وقتی که هیچ محدودیت دیگری روی متغیرها وجود ندارد. به عبارت دیگر، قواعد ساده‌ساز بیان می‌کنند که وقتی یک متغیر در یک مسیر از ریشه به برگ وجود ندارد، چه تفسیری باید صورت گیرد. در دیاگرام تصمیم‌گیری دودویی و دیاگرام تصمیم‌گیری تابعی، این امر نشان می‌دهد که خروجی تابع مستقل از متغیر گره است در حالیکه در دیاگرام تصمیم‌گیری دودویی وقف‌شده به صفر، این مساله نشانگر صفر بودن آن متغیر است.

میناتو نشان داده که می‌توان تعدادی از مسائل ترکیبی را به نحو موثری با دیاگرام تصمیم‌گیری دودویی وقف‌شده به صفر نمایش داد [26]. این مسائل شامل ساده‌سازی منطقی دو سطحی کلاسیک[1] و تکنیک‌های ساده‌سازی چندسطحی[2] می‌گردد. می‌توان نشان داد که دیاگرام تصمیم‌گیری دودویی وقف‌شده به صفر در مقایسه با دیاگرام تصمیم‌گیری دودویی، تعداد گره‌های مورد نیاز برای نمایش یک مجموعه از بردارهای n–بیتی را حداکثر به اندازه n کاهش می‌دهد [29]. در عمل، این اندازه کاهش در تعداد گره‌ها، می‌تواند بسیار موثر باشد.

۲-۷ نمایش توابع چندجمله‌ای

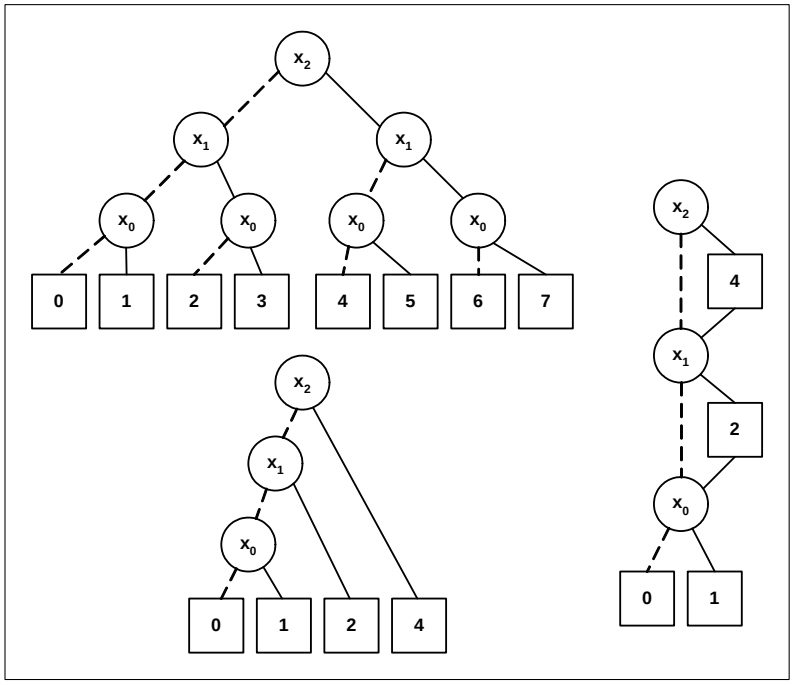
تلاش‌های متعددی صورت گرفت تا با الهام گرفتن از دیاگرام تصمیم‌گیری دودویی و گسترش مفهوم آن، توابع با دامنه بولی و برد غیر بولی (برای مثال: صحیح یا حقیقی) را نمایش داد. این توابع در کاربردهایی مثل محاسبه احتمال فعال شدن یک حالت در مدارهای ترتیبی [10]، روش‌های طیفی[3] برای نگاشت تکنولوژی[4] [17]، و در ارزیابی مدارهای محاسباتی [16][22] به کرات استفاده می‌شوند. از آنجا که متغیرها کماکان بولی هستند، می‌توان

[1] Classical Problems in Two-Level Logic Optimization
[2] Multi-Level Logic Minimization
[3] Spectral Methods
[4] Technology Mapping

از ساختارهای شاخه‌ای[1]، مشابه دیاگرام تصمیم‌گیری دودویی استفاده کرد. مساله اصلی، یافتن یک روش فشرده برای نمایش دادن مقادیر عددی تابع است.

۲-۷-۱ دیاگرام تصمیم‌گیری دودویی با ترمینال‌های متعدد

یک راه ساده برای نمایش توابع چندجمله‌ای، استفاده از یک گراف تصمیم‌گیری مشابه دیاگرام تصمیم‌گیری دودویی می‌باشد، با این تفاوت که ترمینال‌ها می‌توانند هر مقداری را اختیار کنند. به یک چنین نمایشی دیاگرام تصمیم‌گیری دودویی با ترمینال‌های متعدد[2] [17] گفته می‌شود. این ساختمان‌داده همچنین دیاگرام تصمیم‌گیری محاسباتی[3] [10] نیز خوانده می‌شود. شکل ۲-۵ نمایش دیاگرام تصمیم‌گیری دودویی با ترمینال‌های متعدد تابع $x_0 + 2x_1 + 4x_2$ را نشان می‌دهد. این تابع حاصل تفسیر ۳ بیت به عنوان یک عدد دودویی بدون علامت می‌باشد. این نکته قابل ذکر است که ارزیابی دیاگرام تصمیم‌گیری دودویی با ترمینال‌های متعدد برحسب یک ورودی مشخص، مشابه ارزیابی دیاگرام تصمیم‌گیری دودویی است. بر حسب مقدار متغیرها، یک مسیر از ریشه به گره ترمینال را دنبال می‌کنیم و مقدار آن ترمینال را به عنوان خروجی تابع بر می‌گردانیم.



شکل ۲-۵: مثالی از نمایش‌های توابع عددی

[1] Branching Structure
[2] Multi-Terminal Binary Decision Diagram
[3] Arithmetic Decision Diagram

همانطور که این مثال نشان می‌دهد، دیاگرام تصمیم‌گیری دودویی با ترمینال‌های متعدد برای نمایش توابعی که خروجی آنها محدوده بزرگی را می‌پوشاند، مناسب نیستند. برای مثال، اعداد دودویی بدون علامت به طول n، 2^n مقدار متفاوت اختیار می‌کنند و بنابراین نمایش دیاگرام تصمیم‌گیری دودویی با ترمینال‌های متعدد آنها، تعداد نمایی گره ترمینال نیاز خواهد داشت. برای بعضی کاربردها، تعداد خروجی‌های ممکن آنقدر کوچک است که این کاستی جبران می‌شود. برای چنین کاربردهایی، به دلیل سهولت نمایش و شباهت به دیاگرام تصمیم‌گیری دودویی، دیاگرام تصمیم‌گیری دودویی با ترمینال‌های متعدد یک انتخاب مناسب است.

۲-۷-۲ دیاگرام تصمیم‌گیری دودویی با یال‌های وزن‌دار

برای کابردهایی که تعداد خروجی‌های ممکن تابع زیاد است، از روش‌های دیگری برای رسیدن به یک نمایش فشرده استفاده شده است. لای[1] و دیگران، دیاگرام تصمیم‌گیری دودویی با یال‌های وزن‌دار[2] [22][23] را پیشنهاد کردند. در روش آنها به هر یال یک وزن اضافه شد تا اشتراک بیشتری میان زیرگراف‌ها حاصل گردد. به عنوان مثال، نمایش دیاگرام تصمیم‌گیری دودویی با یال‌های وزن‌دار تابع $x_0 + 2x_1 + 4x_2$ در شکل ۲-۵ نشان داده شده است. در این شکل، وزن یالها در مربع‌هایی روی یالها قرار گرفته است و یالهای بدون مربع دارای وزن صفر می‌باشند. ارزیابی یک تابع که بوسیله دیاگرام تصمیم‌گیری دودویی با یال‌های وزن‌دار نمایش یافته است با دنبال کردن مسیری که بوسیله ورودی مشخص می‌شود و جمع کردن وزن یالها و مقدار گره ترمینال انجام می‌گیرد. همانطور که این مثال نشان می‌دهد، مجموع بیت‌های وزن‌دار با قرار دادن وزن هر بیت روی یال خارج‌شونده از گرهی که با آن متغیر برچسب خورده است، نمایش داده می‌شود. این نمایش به صورت خطی برحسب تعداد بیت‌ها افزایش می‌یابد--یک پیشرفت بزرگ نسبت به دیاگرام تصمیم‌گیری دودویی با ترمینال‌های متعدد.

می‌توان روش‌های مختلفی را برای نرمالیزه کردن وزن یالها به کار گرفت، به نحوی که گراف حاصل یک نمایش یگانی برای توابع را فراهم آورد. برای مثال، در دیاگرام تصمیم‌گیری دودویی با یال‌های وزن‌دار استاندارد، وزن یالهای مربوط به مقدار صفر هر متغیر باید صفر باشد. توابع بولی را هم می‌توان بوسیله دیاگرام تصمیم‌گیری دودویی با یال‌های وزن‌دار نمایش داد و تعداد گره‌های مورد نیاز این نمایش برابر با تعداد گره‌های مورد نیاز دیاگرام تصمیم‌گیری دودویی مرتب‌شده متناظر با آن می‌باشد. هرچند در مقایسه با دیاگرام تصمیم‌گیری دودویی مرتب‌شده، نگهداری و دستکاری وزن یالها کارایی آنها را پایین می‌آورد. همچنین، تعداد گره‌های غیر ترمینال در

[1] Lai
[2] Edge-Valued Binary Decision Diagram

دیاگرام تصمیم‌گیری دودویی با یال‌های وزن‌دار همواره از تعداد گره‌های غیر ترمینال در دیاگرام تصمیم‌گیری دودویی با ترمینال‌های متعدد کمتر است البته با هزینه بیشتری در هر یال یعنی وزن یالها.

اگرچه دیاگرام تصمیم‌گیری دودویی با یال‌های وزن‌دار مزایای زیادی نسبت به دیاگرام تصمیم‌گیری دودویی با ترمینال‌های متعدد دارد، اما همچنان دسته‌های مهمی از توابع وجود دارند که به صورت فشرده توسط آنها قابل نمایش نیستند. برای مثال، یکی از کاربردهای مهم توابع عددی، ارزیابی‌رسمی مدارهای محاسباتی از قبیل جمع، ضرب و تقسیم می‌باشد. در این کاربردها از نمایش مقادیر عددی کد شده استفاده می‌شود. به عنوان یک مثال، دو عدد بدون علامت n–بیتی که بوسیله بردارهای بیتی $\vec{x} = x_{n-1},\cdots,x_0$ و $\vec{y} = y_{n-1},\cdots,y_0$ نمایش داده شده‌اند، را در نظر بگیرید. این دو را می‌توان به صورت مقادیر عددی کد شده $X = \sum_{i=0}^{n-1} 2^i x_i$ و به طور مشابه Y در نظر گرفت. همانطور که دیدیم، دیاگرام تصمیم‌گیری دودویی با یال‌های وزن‌دار می‌تواند این توابع کد شده X و Y را با پیچیدگی خطی برحسب n نمایش دهند. حاصل جمع و تفریق این کلمه‌ها ($X+Y$ و $X-Y$) را هم می‌توان به صورت خطی با دیاگرام تصمیم‌گیری دودویی با یال‌های وزن‌دار نمایش داد. اما تعداد گره‌های مورد نیاز این نمایش برای توابع ضرب $X*Y$ و نمایی 2^X به صورت نمایی برحسب n افزایش می‌یابد. بنابراین در ارزیابی مدارهای محاسباتی، دیاگرام تصمیم‌گیری دودویی با یال‌های وزن‌دار به واحدهای کوچک مثل جمع کننده و مقایسه کننده محدود هستند.

۲-۷-۳ دیاگرام‌های گشتاور دودویی

یک روش جایگزین برای نمایش توابع عددی خصوصا آنهایی که در مدارهای محاسباتی به کار می‌روند، تغییر تابع تجزیه (مشابه آنچه در دیاگرام تصمیم‌گیری تابعی انجام گرفت) می‌باشد. نمایش حاصل، دیاگرام گشتاور دودویی[1] [16] نامیده می‌شود.

برای نشان دادن توابع با برد عددی، می‌توان بسط بولی شانون (رابطه ۲-۱) را به صورت زیر تعمیم بخشید:

$$f = (1-x)\cdot f_{\bar{x}} + x\cdot f_x \qquad \textbf{رابطه ۲-٤}$$

که .، + و – بیانگر عملیات ضرب، جمع و تفریق می‌باشند. توجه کنید که این عبارت بر این پیش فرض استوار است که متغیر x بولی می‌باشد یعنی فقط مقادیر 0 و 1 را می‌تواند اختیار کند. هر دوی دیاگرامهای

[1] Binary Moment Diagram

تصمیم‌گیری دودویی با ترمینال‌های متعدد و تصمیم‌گیری دودویی با یال‌های وزن‌دار برحسب چنین تابع تجزیه‌ای بدست می‌آیند.

تجزیه گشتاوری یک تابع با جابجا کردن متغیرها در رابطه ۲-٤، بدست می‌آید:

$$f = f_{\bar{x}} + x \cdot (f_x - f_{\bar{x}}) = f_{\bar{x}} + x \cdot f_{\partial x}$$

رابطه ۲-۵

که $f_{\partial x} = f_x - f_{\bar{x}}$ گشتاور خطی[1] f بر حسب x نامیده می‌شود. این اصطلاح از نگاه خطی به تابع f بر حسب متغیر x ناشی شده است و بنابراین $f_{\partial x}$ مشتق جزئی تابع f بر حسب x می‌باشد. از آنجا که خروجی تابع برای تنها دو مقدار x برای ما اهمیت دارد، همواره می‌توان آنرا به یک فرم خطی گسترش داد. کوفاکتور منفی، گشتاور ثابت[2] هم نامیده می‌شود چرا که بیانگر بخشی از تابع f می‌باشد که برحسب متغیر x ثابت باقی می‌ماند. شباهت میان تجزیه گشتاوری و تجزیه رید-مولر (رابطه ۲-۲) قابل توجه می‌باشد.

هر گره در دیاگرام گشتاور دودویی، یک تابع را بر حسب تجزیه گشتاوری متغیری که به آن گره برچسب خورده است، توصیف می‌کند. به عنوان مثال، نمایش دیاگرام گشتاور دودویی عدد دودویی بدون علامت $x_0 + 2x_1 + 4x_2$ در شکل ۲-۵ نشان داده شده است. دو یال خارج‌شونده از هر گره بیانگر گشتاور ثابت (خط چین) و گشتاور خطی (خط ممتد) تابع برحسب متغیر آن گره می‌باشند. توجه به این نکته ضروری است که گشتاور خطی این مثال برحسب هر متغیر x_i به سادگی 2^i می‌باشد؛ تابع برحسب وزن بیت‌ها تجزیه شده است. این خاصیت، دیاگرام گشتاور دودویی را برای نمایش توابع محاسباتی مناسب می‌سازد.

یک راه گسترش دیاگرام‌های گشتاور دودویی اضافه کردن وزن به یال‌ها می‌باشد. این نمایش دیاگرام گشتاور دودویی ضربی[3] [16] نامیده می‌شود. این وزن‌ها به صورت ضرب‌شونده با هم ترکیب می‌شوند، برخلاف دیاگرام تصمیم‌گیری دودویی با یالهای وزن‌دار که وزن‌ها به صورت جمع‌شونده با هم ترکیب می‌شوند. توابع $X+Y$، $X-Y$، $X \cdot Y$ و 2^X بوسیله دیاگرام گشتاور دودویی ضربی با پیچیدگی خطی قابل نمایش هستند.

هر دو نمایش دیاگرام گشتاور دودویی و دیاگرام گشتاور دودویی ضربی نسبت به سایر نمایش‌های توابع عددی که در این فصل ارائه شده‌اند، مزایای بیشتری دارند. اما این دو، چند نقطه ضعف نیز دارا می‌باشند. بسیاری از عملگرهای آنها در بدترین حالت، پیچیدگی نمایی دارند. به علاوه، دیاگرام گشتاور دودویی می‌تواند توابع بولی را نمایش دهد، اما این نمایش به خوبی دیاگرام تصمیم‌گیری دودویی نیست. در حقیقت، مواردی

[1] Linear Moment
[2] Constant Moment
[3] Multiplicative Binary Moment Diagram

وجود دارد که نمایش دیاگرام گشتاور دودویی یک تابع بولی، بسیار بزرگتر از معادل دیاگرام تصمیم‌گیری دودویی آن می‌باشد.

فصل سوم: دیاگرام بسط تیلور

اخیرا یک نمایش جدید، فشرده، یگانی و مبتنی بر گراف به نام دیاگرام بسط تیلور[1] برای نمایش بهینه توابع جبری معرفی گردیده است. دیاگرام بسط تیلور بر مبنای یک تابع تجزیه جدید و غیربولی ساخته می‌شود. عبارات جبری با استفاده از سری تیلور[2] حول متغیرهای پشتیبان[3] عبارت جبری، شکسته می‌شوند[4]. این بخش‌های شکسته شده را می‌توان به صورت فشرده، یگانی و مبتنی بر گراف نمایش داد. همچنین این مکانیزم غیربولی اجازه می‌دهد که متغیرهای سطح‌کلمه (بردارهای بیتی) را بوسیله علائم[5] جبری مدلسازی کنیم. این امر سطح انتزاع[6] را از بیت به کلمه افزایش می‌دهد. از این رو چنین نمایشی برای ارزیابی‌رسمی در سطح بالا مناسب می‌باشد.

۳-۱ مقدمه

در این فصل یک نمایش جدید، فشرده، یگانی و مبتنی بر گراف به نام دیاگرام بسط تیلور [32][33][34] برای نمایش توابع جبری و بولی معرفی می‌گردد. این ساختمان‌داده را می‌توان برای بررسی برابری[7] در سطح بالا به کار گرفت. به دلیل افزایش پیچیدگی و اندازه طرح‌های دیجیتال[8]، مسائل ارزیابی باید هر چه زودتر در چرخه طراحی[9] انجام گیرند. این امر نیازمند داشتن ابزاری مناسب می‌باشد. به این دلیل، روش‌های[10] ارزیابی‌رسمی از قبیل بررسی خصوصیت[11]، بررسی برابری و شبیه‌سازی به صورت نمادین[12] به بخشی از چرخه طراحی بدل شده‌اند. ابزارهایی برای بررسی برابری طرح‌های سطح پایین (برای مثال سطح گیت) پیاده‌سازی گردیده‌اند. اما ابزار مناسبی برای بررسی برابری طرح‌های سطح بالا وجود نداشته است. بخشی از این امر بدلیل آن است که معمولا توصیف‌ها در سطح انتقال‌ثبات[13] جزئیات پیاده‌سازی واحدهای منطقی را مخفی کرده و مسئولیت را بر عهده سطح انتزاع بعدی می‌گذارند. توصیف‌های سطح بالا بدلیل جزئیات کمتر، کار ارزیابی را ساده‌تر می‌کنند.

---

[1] Taylor Expansion Diagram
[2] Taylor Series
[3] Support Variables
[4] Decompose
[5] Symbols
[6] Abstract
[7] Equivalence Checking
[8] Digital Designs
[9] Design Cycle
[10] Methods
[11] Property Checking
[12] Symbolic
[13] Register Transfer Level

محققان برای رسیدن به نمایشی که بتواند متغیرهای سطح‌کلمه و ارتباط‌شان با متغیرهای بولی را همزمان نشان دهد، مدت‌ها تلاش کردند.

سیستم‌های ارزیابی‌رسمی مانند [35][36] برای ارزیابی واحد کنترل مناسب می‌باشند و ارزیابی‌رسمی واحدهای محاسباتی، معمولا در سطح بیت انجام می‌گیرد [37][38]. تلاش‌هایی برای نمایش متغیرهای سطح‌کلمه صورت گرفته است. برای مثال از روش‌های اثبات قضیه[۱]، روش‌های تصمیم‌گیری خودکار[۲] برای محاسبات پرسبورگر[۳] [39][40] و تکنیک‌های دستکاری جبری[۴] [41][42] می‌توان نام برد. البته هنوز مساله ارزیابی طرح‌های بزرگ – بررسی برابری توصیف‌های انتقال‌ثبات با ارتباط پیچیده میان بخش‌های سطح‌کلمه (محاسباتی) و سطح بیت (بولی) – به طور کامل و جامع حل نشده است.

روش‌های بسیاری برای گسترش قابلیت موتورهای ارزیابی[۵] در نمایش همزمان واحدهای محاسباتی و ارتباط آنها با بخش‌های بولی پیشنهاد شده‌اند. بیشتر این روش‌ها بر مبنای دیاگرام‌های تصمیم‌گیری سطح‌کلمه[۶] می‌باشند. این دیاگرام‌ها امکان نمایش توابع با دامنه بولی و برد عددی را فراهم می‌آورند. برای نمونه می‌توان از دیاگرام تصمیم‌گیری دودویی با ترمینال‌های متعدد[۷] [17][10]، دیاگرام تصمیم‌گیری دودویی با یال‌های وزن‌دار[۸] [54]، دیاگرام گشتاور دودویی ضربی[۹] [16]، دیاگرام گشتاور دودویی ضربی کرانکر[۱۰] [46][47][48] و دیاگرام تصمیم‌گیری ترکیبی[۱۱] [45] نام برد. مرجع شماره [43] دیاگرام‌های تصمیم‌گیری سطح‌کلمه را به خوبی مرور کرده است. اگرچه دیاگرام گشتاور دودویی ضربی و دیاگرام گشتاور دودویی ضربی کرانکر موفقیت‌های چندی در ارزیابی واحدهای محاسباتی کسب کردند، اما آنها برای ارزیابی عبارات چندجمله‌ای درجه بالا بهینه نمی‌باشند. برای نمایش تابع X^3 (X یک بردار بیتی به طول n می‌باشد)، دیاگرام گشتاور دودویی ضربی و دیاگرام گشتاور دودویی ضربی کرانکر به ترتیب به فضایی معادل $O(n^3)$ و $O(n^2)$ نیاز دارند [43].

تابع تجزیه بیشتر دیاگرام‌های تصمیم‌گیری سطح‌کلمه بر مبنای تجزیه توابع بولی می‌باشد. توابع تجزیه مختلف بولی (شانون، دویو، رید–مولر، کرانکر) به دیاگرام‌های سطح‌کلمه مختلفی منجر شده است (با یال‌های وزن‌دار یا

[1] Theorem Proving
[2] Automated Decision Procedure
[3] Presburger Arithmetic
[4] Algebraic Manipulation
[5] Verification Engine
[6] Word Level Decision Diagram
[7] Multi Terminal Binary Decision Diagram
[8] Edge-Valued Binary Decision Diagram
[9] Multiplicative Binary Moment Diagram
[10] Kronecker Multiplicative Binary Moment Diagram
[11] Hybrid Decision Diagram

بدون وزن). برای مثال، با گسترش دیاگرام تصمیم‌گیری دودویی و مجاز دانستن برگ‌های عددی، به دیاگرام تصمیم‌گیری دودویی با ترمینال‌های متعدد (یا دیاگرام تصمیم‌گیری جبری) می‌رسیم. تابع تجزیه در هر گره کماکان بولی است و برگ‌ها مقادیر ثابت عددی را نگهداری می‌کنند.

دیاگرام گشتاور دودویی[1] و مشتقاتش[2]، برخلاف تجزیه بولی دیاگرام تصمیم‌گیری دودویی با ترمینال‌های متعدد، یک تابع خطی را بر حسب گشتاورهایش (گشتاور ثابت و گشتاور خطی) تجزیه می‌کنند. به هر یال گراف یک وزن نسبت داده شده است که به صورت ضربی با هم ترکیب می‌شوند. محدودیت‌های چندی روی وزن یالها اعمال شده تا گراف حاصل یگانی باشد. بسیاری از عملیات محاسباتی (X، $X+Y$، $X.Y$) دارای نمایشی خطی بر حسب تعداد بیت‌های متغیرها می‌باشند. متاسفانه پیچیدگی زمانی این دیاگرام در بدترین حالت رتبه نمایی دارد.

نمایش دیاگرام بسط تیلور [32][33][34] بر اساس یک تابع تجزیه جدید می‌باشد. در این نمایش، عبارت‌ها به صورت توابع پیوسته و مشتق‌پذیر در نظر گرفته می‌شوند. همچنین تجزیه هر متغیر سطح‌کلمه بوسیله بسط تیلور انجام می‌گیرد. در هر سطح، عبارت برحسب یکی از متغیرها تجزیه می‌شود. در نمایش توابع چندجمله‌ای، تعداد بخش‌های تجزیه شده در هر سطح محدود بوده و برابر با درجه متغیر مربوط به آن سطح می‌باشد. تعداد گره‌های مورد نیاز برای نمایش عبارات جبری متداول در توصیفات انتقال‌ثبات (برای مثال $X+Y$، $X-Y$، X^k) رابطه‌ای خطی با تعداد متغیرها دارد. در حقیقت دیاگرام بسط تیلور اولین نمایشی بوده که متغیرهای با درجه بزرگتر از دو را به صورت خطی نمایش می‌دهد. بعلاوه این دیاگرام می‌تواند توابع جبری و بولی را همزمان نمایش دهد (همانطور که دیدیم همه دیاگرام‌های سطح‌کلمه این ویژگی را دارا می‌باشند).

۳-۲ دیاگرام بسط تیلور

در این نمایش متغیرها به شکل اعداد حقیقی[3] (در عمل به شکل اعداد صحیح[4]) فرض می‌شوند. فرض کنید که $f(x,y,\ldots)$ یک تابع حقیقی و مشتق‌پذیر باشد. با اعمال بسط تیلور روی متغیر x و با فرض مقدار اولیه صفر خواهیم داشت:

$$f(x)=f(0)+xf'(0)+\frac{1}{2}x^2 f''(0)+\cdots$$ **رابطه ۳-۱**

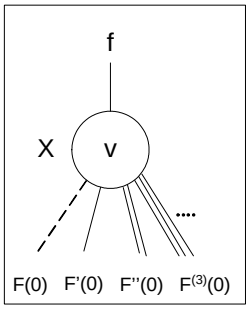
[1] Binary Moment Diagram
[2] Derivation
[3] Real
[4] Integer

که $f'(0)$ و $f''(0)$ مشتق اول و دوم تابع f بر حسب متغیر x می‌باشند. این مشتقات مستقل از متغیر x می‌باشند، لذا می‌توان آنها را برحسب سایر متغیرها و بوسیله بسط تیلور تجزیه کرد (متغیر به متغیر). بخش‌های تجزیه شده را می‌توان به صورت یک دیاگرام فشرده نمایش داد. این ساختمان‌داده، دیاگرام بسط تیلور نام گرفت.

۳-۲-۱ ساختار دیاگرام بسط تیلور

دیاگرام بسط تیلور یک گراف جهت‌دار[1] و بدون دور[2] (Φ, V, E, T) می‌باشد که یک عبارت جبری را نمایش می‌دهد. Φ عبارت جبری نمایش یافته بوسیله دیاگرام بسط تیلور می‌باشد. V مجموعه گره‌ها و E مجموعه یالهای وزن‌دار می‌باشند. T مجموعه گره‌های ترمینال می‌باشد. گره ریشه، تابع Φ را نمایش می‌دهد. یالهای خارج‌شونده از این گره به مشتقات تابع (گره‌های فرزند) بر حسب متغیر ریشه اشاره می‌کنند. این مکانیزم تجزیه به صورت بازگشتی به همه گره‌های فرزند هم اعمال می‌گردد و دیاگرام بسط تیلور حاصل می‌شود.

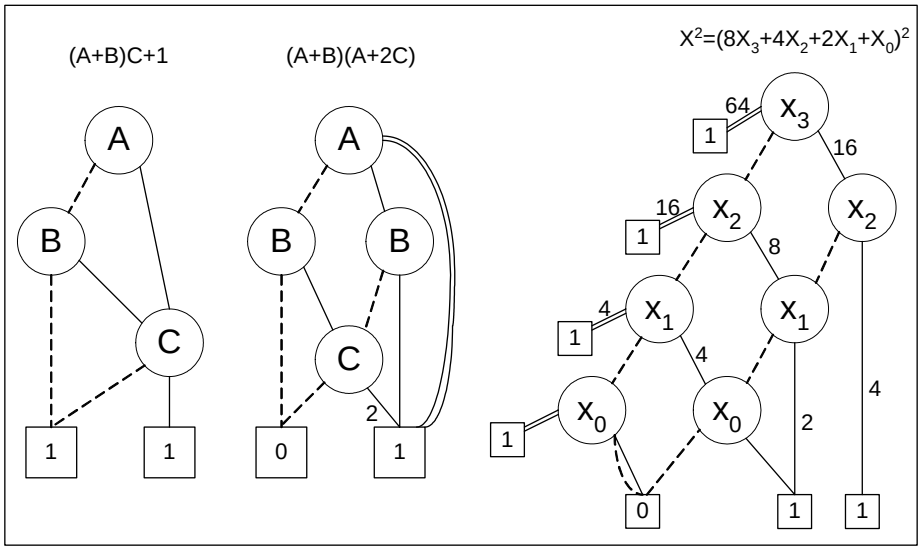
هر گره v یک برچسب[3] (شاخص[4]) دارد ($var(v)$) که متغیر آن گره را مشخص می‌کند. مثل سایر دیاگرام‌ها، بین متغیرهای دیاگرام بسط تیلور ترتیب مشخصی وجود دارد. تابع هر گره $v \in V$ از روی بسط تیلور و مطابق رابطه ۳-۱ بدست می‌آید. تناظر یک به یکی میان گره‌های داخلی[5] و مشتقات متوالی تابع $f(x)$ در نقطه صفر وجود دارد ($f(0)$ ، $f'(0)$ ، $f''(0)$). تعداد گره‌های خارج‌شونده از هر گره به درجه متغیر آن گره بستگی دارد. تعداد یال‌های خارج‌شونده از گره‌های ترمینال صفر می‌باشد. تابع نمایش یافته توسط یک گره ترمینال یک مقدار عددی ثابت می‌باشد.



شکل ۳-۱: تجزیه در دیاگرام بسط تیلور

[1] Directed
[2] Acyclic
[3] Label
[4] Index
[5] Internal

شکل ۳-۱ تجزیه تابع f را بر حسب متغیر x نشان می‌دهد. به مشتق k–ام یک تابع در گره v با $var(v)=x$، فرزند k–ام آن گره گفته می‌شود: $f(x=0)$ فرزند–۰ ، $f'(x=0)$ فرزند–۱، $f''(x=0)$ فرزند–۲. همچنین به یال‌های متناظر یال–۰ (خط چین)، یال–۱ (خط ممتد)، یال–۲ (خط دوگانه) گفته می‌شود. توجه کنید که به هر یال یک وزن تلویحی[1] نسبت داده شده است (چرا که فقط مشتقات تابع دارای ارزش عملی بوده و باید نگهداری شوند و ضرائب مشتقات در سری تیلور برای همه توابع یکسان بوده و به عنوان وزن تلویحی به یال‌ها نسبت داده می‌شوند): $x^0=1$ برای یال–۰، $x^1=1$ برای یال–۱ و $\frac{x^2}{2}$ برای یال–۲. همچنین به یال‌های گراف، وزن‌های ضرب‌شونده‌ای نسبت داده شده است که از روی سری تیلور محاسبه می‌گردد. شکل ۳-۲ نمایش دیاگرام بسط تیلور چند عبارت جبری ساده را نشان می‌دهد.



شکل ۳-۲: نمایش چند عبارت جبری با دیاگرام بسط تیلور

محاسبه مشتقات (و در نتیجه فرزندهای f) برای توابع چندجمله‌ای کار ساده‌ای می‌باشد. عبارات پیچیده را می‌توان به صورت ترکیبی از عبارات ساده‌تر بدست آورد. در زیر به توضیح عملگرهای ضرب و جمع خواهیم پرداخت. توجه به این نکته ضروری است که عملگر تفریق را می‌توان با استفاده از عملگرهای جمع و ضرب در ۱– بدست آورد.

[1] Implicit

۳-۲-۲ عملگرهای جمع و ضرب

روش ساختن دیاگرام بسط تیلور مشابه دیاگرام تصمیم‌گیری دودویی به صورت بازگشتی[1] می‌باشد. اما قواعد ترکیب[2] دیاگرام بسط تیلور متفاوت است چرا که باید مطابق قواعد جبری $(R,+,*)$ باشد.

برای ساختن دیاگرام بسط تیلور حاصل از یک عمل خاص (مثل جمع یا ضرب)، از ریشه دیاگرام بسط تیلور مربوط به عملوندها شروع کرده و به صورت بازگشتی همه بخش‌های غیرصفر آن دو را ساخته و به نحو مناسب با هم ترکیب می‌کنیم تا دیاگرام تابع جدید بدست آید. برای اطمینان از اینکه گراف حاصل به ساده‌ترین شکل ممکن درآمده باشد، عملگر ساده‌ساز را به آن اعمال می‌کنیم. در مورد این عملگر در بخش ۳-۳ توضیح داده خواهد شد.

فرض کنید u و v دو گره هستند که باید با هم ترکیب شوند و گره q حاصل این ترکیب می‌باشد. همچنین فرض کنید که $var(u)=x$ و $var(v)=y$ نشانگر متغیرهای وابسته به این دو گره باشند. همانطور که بعداً توضیح داده خواهد شد، $var(q)$ برابر با متغیر کوچکتر خواهد بود. بدین معنا که $var(q)=x$ اگر $y \geq x$ و در غیر این صورت $var(q)=y$. فرض کنید f و g دو تابع هستند که بوسیله گره‌های u و v نمایش یافته‌اند. همچنین h تابعی است که بوسیله گره q نمایش یافته است. نهایتاً T مجموعه همه گره‌های ترمینال می‌باشد. اگر $T \in v$ باشد، آنگاه $val(v)$ مقدار آن گره را نشان می‌دهد.

برای سادگی در بررسی زیر فقط یال-۰ و یال-۱ برای هر گره در نظر گرفته شده است. اما این تحلیل برای گره‌هایی با فرزندان متعدد هم صادق می‌باشد. در این تحلیل موارد زیر در نظر گرفته شده است.

۱. هر دو گره ترمینال باشند، $T \in u, v$. در این حالت یک گره ترمینال q به صورت زیر ساخته می‌شود.

جمع: $val(q)=val(u)+val(v)$

ضرب: $val(q)=val(u)*val(v)$

۲. اگر یکی از دو گره ترمینال نباشد، عملیات زیر برحسب ترتیب متغیرها انجام می‌گیرد.

اگر دو گره با یک متغیر برچسب خورده باشند، آنگاه $var(q)=x$.

جمع:

$$h(x) = f(x) + g(x) = f(0) + xf'(0) + g(0) + xg'(0) = [f(0) + g(0)] + x[f'(0) + g'(0)]$$

رابطه ۳-۲

[1] Recursive
[2] Composition Rules

یعنی فرزندان متناظر دو دیاگرام با هم جمع می‌شوند.

ضرب:

رابطه ۳-۳

$$h(x) = f(x).g(x) = (f(0) + xf'(0)).(g(0) + xg'(0)) =$$
$$[f(0).g(0)] + x[f(0).g'(0) + f'(0).g(0)] + x^2[f'(0).g'(0)]$$

یعنی فرزند-۰ q، از حاصل‌ضرب فرزندان-۰ u و v حاصل می‌شود. فرزند-۱ حاصل‌ضرب از حاصل‌جمع دو حاصل‌ضرب متقاطع[1] فرزندان-۰و۱ حاصل شده و بنابراین به یک عمل جمع اضافی نیاز دارد. همچنین یک فرزند-۲ هم حاصل می‌شود که از حاصل‌ضرب فرزندان-۱ u و v بدست می‌آید.

۳. اگر دو گره با متغیرهای مختلفی برچسب خورده باشند، آنگاه $var(q)=\max(var(u),var(v))$. اگر $ord(x)>ord(y)$ باشد، خواهیم داشت:

جمع:

رابطه ۳-٤

$$h(x) = f(x) + g(y) = f(x=0) + xf'(x=0) + g(y) =$$
$$[f(x=0) + g(y)] + xf'(x=0)$$

یعنی $g(y)$ از گره v به فرزند-۰ گره u اضافه می‌شود و فرزند-۱ گره u بدون تغییر باقی می‌ماند.

ضرب:

رابطه ۳-۵

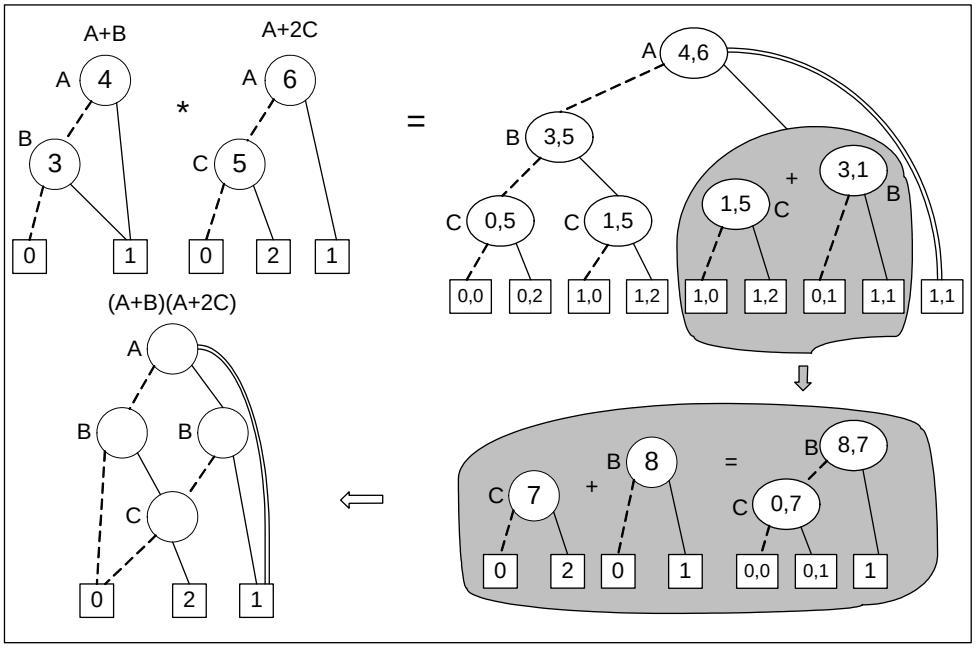
$$h(x) = f(x).g(x) = (f(x=0) + xf'(x=0)).g(y) =$$
$$[f(x=0).g(y)] + x[f'(x=0).g(y)]$$

در اینجا، همه فرزندان گره u در $g(y)$ از گره v ضرب می‌شوند.

توجه به این نکته ضروری است که این مورد، همچنین موردی را که یکی از گره‌ها (در این مورد v) ترمینال باشد، پوشش داده است (g ثابت می‌باشد).

[1] Cross Products

شکل ۳-۳ چگونگی ضرب دو دیاگرام بسط تیلور را نشان می‌دهد.



شکل ۳-۳: نمایش دیاگرام بسط تیلور حاصل‌ضرب (A+B) در (A+2C)

۳-۳ ویژگی‌های نمایش دیاگرام بسط تیلور

در این بخش نحوه ساختن دیاگرام بسط تیلور ساده‌شده[1] بررسی خواهد شد. همچنین پیچیدگی نمایش دیاگرام بسط تیلور بررسی خواهد گردید.

با حذف گره‌های اضافی و ترکیب بخش‌های مشابه، دیاگرام بسط تیلور به ساده‌ترین حالت ممکن تبدیل می‌شود (دیاگرام بسط تیلور ساده‌شده).

تعریف ۱:

یک گره اضافی است اگر که

۱) همه یال‌های خارج‌شونده از آن گره به ترمینال صفر اشاره کنند. یا

۲) آن گره فقط یک بخش ثابت (یال-۰) داشته باشد.

در هر دو مورد گره اضافی حذف و همه اشاره‌گرهایی که به آن اشاره می‌کنند به فرزند-۰ آن گره اشاره داده می‌شوند.

[1] Reduced TED

همچنین دو دیاگرام بسط تیلور شبیه هم[1] هستند اگر ساختار[2] و خواص[3] مشابهی داشته باشند. یا به عبارت دقیقتر:

تعریف ۲:

دو گراف بسط تیلور G و G' مشابه می‌باشند اگر نگاشت یک به یک σ از گره‌های G به گره‌های G' وجود داشته باشد به نحوی که برای هر گره $v \in G$ اگر $\sigma(v) = v' \in G'$ برقرار باشد آنگاه یا

- هر دو گره v و v' ترمینال بوده و $value(v) = value(v')$. یا
- هیچ کدام از گره‌ها ترمینال نبوده، هر دو با یک متغیر برچسب خورده باشند $\text{var}(v) = \text{var}(v')$ ، همه فرزندهای آن دو مشابه باشند $\sigma(child(edge\_k(v))) = child(edge\_k(v'))$ و یال‌های متناظر وزن‌های یکسان داشته باشند.

تعریف ۳:

یک دیاگرام بسط تیلور به ساده‌ترین شکل ممکن نشان داده شده است اگر هیچ گره اضافی نداشته باشد و همچنین هیچ دو گره متفاوتی نداشته باشد که زیرگراف‌های آنها مشابه باشند.

تعریف ٤:

یک دیاگرام بسط تیلور مرتب‌شده می‌باشد اگر در هر مسیر از ریشه به گره‌های ترمینال ترتیب ظاهر شدن متغیرها یکسان باشد و هر متغیر فقط یکبار در هر مسیر ظاهر شود.

یک دیاگرام بسط تیلور که به ساده‌ترین شکل ممکن در آمده است و از یک ترتیب مشخص میان متغیرها تبعیت می‌کند، یگانی می‌باشد.

۳-٤ پیچیدگی نمایش بسط تیلور

در نمایش عبارات خطی، پیچیدگی فضایی[4] دیاگرام بسط تیلور مشابه دیاگرام گشتاور دودویی ضربی، خطی می‌باشد. همچنین دیاگرام بسط تیلور عمل ضرب $(X.Y)$ را مستقل از تعداد بیت‌های دو عملوند آن به صورت

[1] Isomorphic
[2] Structure
[3] Attributes
[4] Space Complexity

خطی نمایش می‌دهد، چرا که حاصل‌ضرب دو یا چند بردار بیتی مختلف به چندجمله‌ای خطی منجر می‌شود. برای نمایش چندجمله‌ای‌هایی با درجه $k \geq 2$، اندازه دیاگرام گشتاور دودویی ضربی به صورت $O(n^k)$ گسترش پیدا می‌کند (n تعداد بیت‌ها می‌باشد). همچنین دیاگرام گشتاور دودویی ضربی کرانکر هم در نمایش چندجمله‌ای‌های با درجه ۳ یا بزرگتر، به صورت $O(n^{k-1})$ گسترش می‌یابد. شایان ذکر است که دیاگرام بسط تیلور همه این عبارت‌ها را به صورت خطی بر حسب تعداد متغیرها نمایش می‌دهد. قضیه زیر این موضوع را اثبات می‌کند.

قضیه:

فرض کنید که متغیر X یک بردار بیتی به طول n می‌باشد: $X = \sum_{i=0}^{n-1} 2^i x_i$. تعداد گره‌های دیاگرام بسط تیلور مورد نیاز برای نمایش X^k برابر $k(n-1)+1$ می‌باشد.

اثبات:

در بدو امر قضیه را برای حالت درجه دوم $k = 2$ اثبات می‌کنیم. فرض کنید که W_m نمایش m-بیتی X باشد: $X = W_n = \sum_{i=0}^{n-1} 2^i x_i = 2^{n-1} x_{n-1} + W_{n-1}$ که $W_{n-1} = \sum_{i=0}^{n-2} 2^i x_i$، $n-1$ بیت کم ارزش X را در بر دارد. بنابراین رابطه $W_n^2 = (2^{n-1} x_{n-1} + W_{n-1})^2 = 2^{2(n-1)} x_{n-1}^2 + 2^n x_{n-1} W_{n-1} + W_{n-1}^2$ برقرار خواهد بود. همچنین خواهیم داشت: $W_{n-1} = (2^{n-2} x_{n-2} + W_{n-2})$ و $W_{n-1}^2 = (2^{2(n-2)} x_{n-2}^2 + 2^{n-1} x_{n-2} W_{n-2} + W_{n-2}^2)$.

توجه کنید که بخش ثابت (یال-۰) W_{n-1} بر حسب متغیر x_{n-2} برابر W_{n-2} می‌باشد. بخش خطی (یال-۱) W_{n-1}^2 برابر است با $2^{n-1} W_{n-2}$. این بدان معنا است که W_{n-2} را می‌توان بین این دو به اشتراک گذاشت. در نتیجه، تنها دو عامل غیر ثابت W_{n-2} و W_{n-2}^2 در این سطح وجود خواهد داشت.

در حالت کلی، سطح l با متغیر x_{n-l} برچسب خورده است و بسط W_{n-l}^2 و W_{n-l}، دقیقا دو عامل غیر ثابت تولید می‌کند: W_{n-l-1}^2 و W_{n-l-1}. عامل W_{n-l-1} را می‌توان با ضرائب ثابت مختلف بین W_{n-l-1}^2 و W_{n-l-1} به اشتراک گذاشت.

به سادگی می‌توان این استدلال را به هر توان دلخواه k تعمیم داد: همیشه در هر سطح k عامل غیر ثابت وجود خواهد داشت. از آنجا که بالاترین متغیر[1] (x_{n-1}) یکی می‌باشد (ریشه) و دقیقا k عامل غیر ثابت در $n-1$ سطح باقیمانده وجود خواهد داشت، تعداد کل گره‌ها برابر $k(n-1)+1$ خواهد گردید.

[1] Top Variable

۳-۵ نمایش توابع بولی و ارتباط بولی-جبری در دیاگرام بسط تیلور

همانطور که گفته شد، دیاگرام بسط تیلور تابع مورد نظر را به صورت یک تابع حقیقی و مشتق‌پذیر در نظر می‌گیرد. در نتیجه واضح است که اصولا چنین نمایشی مناسب توابع جبری می‌باشد (توابع چندجمله‌ای). اگر چه دیاگرام بسط تیلور برای بررسی برابری دو معادله جبری پیچیده مناسب می‌باشد، اما هدف اصلی آن است که دیاگرام بسط تیلور را برای بررسی برابری دو طرح واقعی در سطح انتقال‌ثبات به کار برد. این طرح‌ها معمولا شامل بخش‌های محاسباتی (واحدهای محاسباتی) و بخش‌های بولی (در کنار واحدهای محاسباتی و در واحد کنترل) می‌باشند. بنابراین دیاگرام بسط تیلور باید بتواند توابع بولی و ارتباط آن با توابع جبری را نمایش دهد.
در این بخش نشان خواهیم داد که چگونه می‌توان از دیاگرام بسط تیلور برای نمایش توابع بولی استفاده کرد. بیاد آورید که دیاگرام بسط تیلور، توابع با دامنه و برد صحیح را نمایش می‌دهد. اگر دامنه نمایش دیاگرام بسط تیلور را به {۰و۱} محدود کنیم، آنگاه نمایش توابع بولی کار ساده‌ای خواهد بود. البته باید دقت داشت که در این حالت تغییرات اندکی در نحوه انجام بعضی عملگرها بوجود می‌آید. این موضوع در زیر مورد بحث قرار گرفته است.

۳-۵-۱ نمایش توابع بولی با دیاگرام‌های بسط تیلور

در زیر عملگرهای دیاگرام بسط تیلور برای توابع بولی تعریف می‌گردند. دامنه و برد این توابع بولی می‌باشد. مشتق این عملگرها باید بر اساس قواعد جبری $R(+,*)$ صورت بگیرد. مشابه آنچه در دیاگرام گشتاور دودویی ضربی صورت گرفته است، از روابط زیر می‌توان به عنوان عملگرهای بولی دیاگرام بسط تیلور استفاده کرد.

$$NOT(x)=x'=(1-x) \qquad \textbf{رابطه ۳-۶}$$

$$AND(x,y)=x\wedge y=x*y \qquad \textbf{رابطه ۳-۷}$$

$$OR(x,y)=x\vee y=x+y-x*y \qquad \textbf{رابطه ۳-۸}$$

هر مداری که منحصرا از بخش‌های بولی تشکیل یافته باشد را به سادگی می‌توان بوسیله روابط بالا به دیاگرام بسط تیلور تبدیل کرد.

۳-۵-۲ ارتباط بخش‌های جبری و بولی

می‌دانیم که نمایش دیاگرام بسط تیلور مدارهای منحصرا بولی، همیشه به جواب درستی منجر می‌شود. آنچه که باقی می‌ماند آنست که نشان دهیم ارتباط متغیرهای جبری و بولی به نمایش درستی منجر می‌شود.

می‌توان ادعا کرد که تنها موردی که باید مورد توجه قرار گیرد، متغیرهای بولی x^k با $k \geq 2$ می‌باشد (مثل $F=G+x+x^2$، G یک عبارت جبری و $x \in \{0,1\}$ می‌باشند). چرا که آنها تنها مواردی هستند که احتمالا به محاسبه اشتباه مقادیر جبری منجر می‌شوند. دلیل این امر آن است که این توابع جبری بوده و محاسبات آنها بر حسب جبر $(R,+,*)$ صورت می‌گیرد، حال آنکه متغیرها می‌توانند هم در سطح‌کلمه و هم در سطح بیت $\{$۱و۰$\}$ باشند.

تابع جمع هرگز جواب اشتباه تولید نمی‌کند. مثلا $F=G+x+x=G+2x$ همیشه به جواب صحیح منجر می‌شود، هم برای $x \in R$ و هم برای $x \in \{0,1\}$. اما تابع ضرب می‌تواند توان چندم یک متغیر بولی را ایجاد کند، x^k، و بنابراین به صورت بالقوه[1] می‌تواند به اشکالاتی در ارزیابی تابع منجر شود. به علاوه، این اشکال معمولا برای متغیرهایی که به بسط‌شان مرتبط‌اند، اتفاق می‌افتد. برای مثال، هنگام ضرب کردن یک متغیر سطح‌کلمه $X=\sum_{i=0}^{n-1} 2^i x_i$ و متغیر بولی x_l $(l \leq n-1)$. توجه کنید که برای $x \in \{0,1\}$، $x=x^2=\cdots=x^k$. بنابراین در این موارد می‌بایست که همه نمونه‌های x^k را با x جایگزین کرد. این امر امکان می‌دهد که ارزیابی مقادیر توابع جبری به درستی صورت گیرد. با این اصلاح[2]، ارتباط متغیرهای جبری-بولی به شکل زیر صورت می‌گیرد.

$$h(x)=f(x).g(x)=(f(0)+xf'(0)).(g(0)+g'(0))= \\ [f(0)g(0)]+x[f(0)g'(0)+f'(0)g(0)+f'(0)g'(0)]$$

رابطه ۳-۹

می‌توان نشان داد که دیاگرام بسط تیلور اصلاح‌شده، یگانی بوده و می‌تواند مستقیما برای بررسی برابری طرح‌های انتقال‌ثبات مورد استفاده قرار گیرد.

[1] Potentially
[2] Modification

فصل چهارم: دیاگرام دودویی تیلور

در این فصل یک روش بهینه برای پیاده‌سازی دیاگرام بسط تیلور ارائه می‌گردد. دیاگرام دودویی تیلور[1] همانند دیاگرام بسط تیلور[2] برحسب سری تیلور ساخته می‌شود، اما از یک ساختمان‌داده دودویی برای نمایش سری تیلور استفاده می‌کند. این شیوه همچنین با ساختارهای متداول قبلی سازگار بوده و در نتیجه برای بررسی برابری[3] بین دو مدار مناسب می‌باشد. پیچیدگی زمانی الگوریتم‌های دیاگرام دودویی تیلور کمتر از دیاگرام بسط تیلور می‌باشد و بنابراین دیاگرام دودویی تیلور به زمان کمتری برای اجرا نیاز دارد.

٤-١ مقدمه

امروزه پیچیدگی سخت‌افزارها افزایش پیدا کرده و در نتیجه فرآیند ارزیابی مهمتر و اساسی‌تر گردیده است. بهتر است ارزیابی یک طرح هر چه زودتر انجام گیرد چرا که هزینه ارزیابی در سطوح پایین‌تر بسیار بیشتر از هزینه ارزیابی در سطوح بالا می‌باشد. بنابراین به یک روش مطمئن و سطح بالا برای ارزیابی طرح‌ها نیاز می‌باشد. ارزیابی‌رسمی یک روش مطمئن بوده که در دهه‌های گذشته پیشنهاد شده و مکمل مناسبی برای شبیه‌سازی[4] می‌باشد. شبیه‌سازی برای طرح‌های بزرگ نامناسب می‌باشد چرا که امکان ندارد بتوان یک طرح بزرگ را برای همه ورودی‌های ممکن در یک زمان قابل قبول شبیه‌سازی کرد. ارزیابی‌رسمی مسائل شبیه‌سازی را با اعمال قواعد ریاضی برای اثبات صحت پیاده‌سازی برطرف می‌نماید.

دو شیوه ارزیابی‌رسمی، بررسی برابری و بررسی خصوصیت[5] می‌باشند. برای بررسی برابری، توابع جبری، بولی و رابطه‌ها را بوسیله یک ساختمان‌داده مناسب نمایش می‌دهند. تاکنون ساختمان‌داده‌های دیاگرام تصمیم‌گیری دودویی[6] [13]، دیاگرام تصمیم‌گیری تابعی[7] [21]، دیاگرام تصمیم‌گیری تابعی کرانکر[8] [44]، دیاگرام تصمیم‌گیری ترکیبی[9] [45]، دیاگرام گشتاور دودویی[10] [16]، دیاگرام گشتاور دودویی ضربی کرانکر[11] [46][47][48] و دیاگرام بسط تیلور[12] [32][33][34] پیشنهاد داده شده‌اند. همه این ساختمان‌های داده مبتنی بر گراف بوده و یگانی می‌باشند.

---

[1] Binary Taylor Diagram
[2] Taylor Expansion Diagram
[3] Equivalence Checking
[4] Simulation
[5] Property Checking
[6] Binary Decision Diagram
[7] Functional Decision Diagram
[8] Kronecker Functional Decision Diagram
[9] Hybrid Decision Diagram
[10] Binary Moment Diagram
[11] Kronecker Multiplicative Binary Moment Diagram
[12] Taylor Expansion Diagram

دیاگرام تصمیم‌گیری دودویی ساده‌شده، اولین نمایش مبتنی بر گراف و یگانی توابع بولی می‌باشد و توسط برایانت[1] [13] ارائه گردید. این دیاگرام یک گراف دودویی جهت‌دار بوده که بر حسب تجزیه شانون[2] عمل می‌نماید.

$$f^{v}(x_1, x_2, ..., x_n) = [\neg x_i \wedge f^{low(v)}(x_1, x_2, ..., x_n)] \vee [x_i \wedge f^{high(v)}(x_1, x_2, ..., x_n)] \qquad \textbf{رابطه ٤-١}$$

که

$$f^{low(v)}(x_1, x_2, ..., x_n) = f(x_1, x_2, ..., x_n)|_{x_i=0} \qquad \textbf{رابطه ٤-٢}$$

$$f^{high(v)}(x_1, x_2, ..., x_n) = f(x_1, x_2, ..., x_n)|_{x_i=1} \qquad \textbf{رابطه ٤-٣}$$

و $\mathrm{var}(v) = x_i$.

برای آنکه بتوان علاوه بر توابع بولی، توابع جبری را هم نمایش داد، دیاگرام‌های تصمیم‌گیری سطح‌کلمه[3] معرفی شدند. این دیاگرام‌ها از مقادیر عددی بجای مقادیر بولی و از عملگرهای + و × بجای & و | استفاده می‌کنند. آخرین دیاگرام پیشنهاد شده (دیاگرام بسط تیلور)، از سری تیلور به عنوان تابع تجزیه استفاده کرده و توابع جبری با درجه بیشتر از یک را پشتیبانی می‌نماید. برخلاف سایر روش‌ها، دیاگرام بسط تیلور بر مبنای یک گراف غیر دودویی ساخته می‌شود. تعداد فرزندان هر گره در دیاگرام بسط تیلور به درجه متغیر آن گره بستگی دارد. بنابراین دیاگرام بسط تیلور از یک ساختمان‌داده پیچیده‌تر نسبت به بقیه روشها، استفاده می‌کند.

در دهه‌های گذشته، چندین الگوریتم بهینه برای پیاده‌سازی ساختمان‌های داده دودویی پیشنهاد داده شده است [12][49][50]. نمایش پیشنهادی ما، برخلاف دیاگرام بسط تیلور از یک ساختمان‌داده دودویی ساده استفاده می‌کند. بنابراین می‌توان از این الگوریتم‌ها برای پیاده‌سازی یک بسته دیاگرام دودویی تیلور بهینه استفاده کرد.

٤-٢ دیاگرام دودویی تیلور

دیاگرام دودویی تیلور یک درخت دودویی برای نمایش توابع جبری و بولی بوده و تابع تجزیه آن سری تیلور می‌باشد.

[1] Bryant
[2] Shannon
[3] Word-Level Decision Diagram

٤-٢-١ سری تیلور

سری تیلور می‌تواند همه توابعی که بینهایت مشتق‌پذیر هستند (لگاریتمی، نمایی یا سینوسی) را نمایش دهد. بنابراین تجزیه برحسب تیلور اجازه می‌دهد که بتوان بسیاری از طرح‌ها را در سطوح بالای انتزاع نمایش داد. برای مثال تابع لگاریتمی را می‌توان بوسیله یک رشته محدود از سری تیلور و با دقت قابل قبول نمایش داد. در سطح ساختاری بیشتر قسمت‌های طرح را توابع چندجمله‌ای تشکیل می‌دهند. در حالت کلی ضرائب چندجمله‌ای، اعداد حقیقی هستند اما در بیشتر طرح‌های واقعی ضرایب صحیح کافی می‌باشند.

سری تیلور تابع مشتق‌پذیر $f(x)$ حول $x=0$ برابر است با

$$f(x) = f(0) + xf'(0) + \frac{1}{2!}x^2 f''(0) + \frac{1}{3!}x^3 f'''(0) + \ldots$$ **رابطه ٤-٤**

که $f'(0)$، $f''(0)$ و $f'''(0)$ مشتقات درجه اول، دوم و سوم تابع f حول $x=0$ می‌باشند. با فاکتور گرفتن از x، رابطه ٤-٤ به صورت زیر در می‌آید.

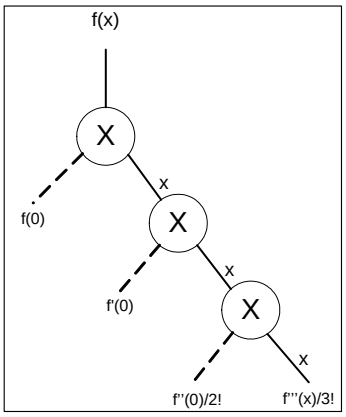
$$f(x) = f(0) + x(f'(0) + x(\frac{1}{2!}f''(0) + x(\frac{1}{3!}f^{(3)}(0) + \ldots)))$$ **رابطه ٤-٥**

تجزیه یک تابع به صورت بازگشتی مطابق رابطه ٤-٥ صورت می‌گیرد.

٤-٢-٢ ساختن دیاگرام دودویی تیلور

دیاگرام دودویی تیلور نمایشی برای توابع جبری و بولی چندمتغیره می‌باشد. این دیاگرام یک گراف دودویی جهت‌دار وزن‌دار می‌باشد و همه گره‌های آن یک برچسب دارند که متغیر آن گره را مشخص می‌نماید. مانند بسیاری از دیاگرام‌های تصمیم‌گیری، متغیرهای دیاگرام دودویی تیلور مرتب‌شده می‌باشند. قرار دادن ترتیب میان متغیرها یک قاعده اساسی برای رسیدن به خاصیت یگانی می‌باشد.

برای نمایش تابع $f(x)$ به صورت دیاگرام دودویی تیلور، رابطه ٤-٥ به گره ریشه اعمال می‌گردد. گره ریشه دو فرزند دارد: $f(0)$ فرزند چپ و $(f(x)-f(0))/x$ فرزند راست. همه گره‌های میانی هم دو فرزند دارند. فرزند چپ با قرار دادن صفر بجای متغیر تجزیه در تابع مربوط به گره پدر ساخته می‌شود. فرزند راست با رابطه $(f(x)-f(0))/x$ که $f(x)$ تابع گره پدر و x متغیر تجزیه گره پدر می‌باشد بدست می‌آید (شکل ٤-١). درجه خروجی همه گره‌ها دو بوده و درجه خروجی گره‌های ترمینال صفر می‌باشد.

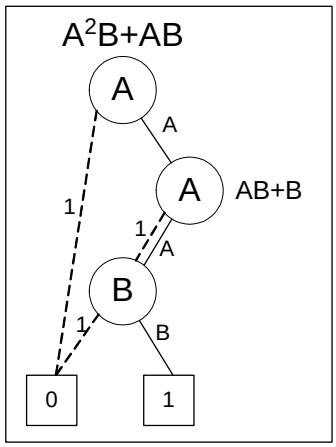


شکل ٤-١: تجزیه گره در دیاگرام دودویی تیلور

فقط دو گره ترمینال معتبر وجود دارد، 0 و 1. همه گره‌های غیر صفر را می‌توان بوسیله ترمینال 1 و یک وزن ضربی روی یال ورودی ساخت. برای رسیدن به خاصیت یگانی، بزرگترین مقسوم‌علیه مشترک وزن یال‌های خروجی هر گره به یال ورودی آن گره منتقل می‌شود. بنابراین وزن یال‌های خروجی هر گره نسبت به هم اول می‌باشند. همه ترمینال‌های 1 را با هم ترکیب می‌کنیم تا تعداد گره‌ها کمتر شود. این کار برای ترمینال‌های 0 هم انجام می‌شود.

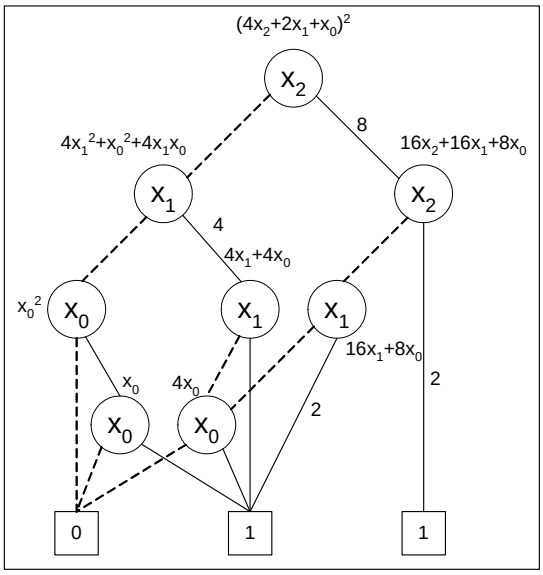
برای بدست آوردن تابع جبری یک دیاگرام دودویی تیلور، همه مسیرهای از ریشه به ترمینال 1 را می‌پیماییم. در این مسیرها و از گره ترمینال به ریشه، در هر گره، تابع فرزند راست را در متغیر آن گره ضرب کرده و با تابع فرزند چپ جمع می‌نمائیم. نهایتا این حاصل را در وزن ورودی این گره ضرب می‌نمائیم.

مثال ١: شکل ٤-٢ نمایش دیاگرام دودویی تیلور عبارت ساده جبری (A^2B+AB) را نشان می‌دهد.



شکل ٤-٢: نمایش دیاگرام دودویی تابع A^2B+AB

مثال ۲: نمایش $X^2=(4x_2+2x_1+x_0)$ در شکل ٤-۳ نشان داده شده است، که X عدد صحیح سه بیتی با بیت‌های x_0، x_1 و x_2 می‌باشد $(x_2>x_1>x_0)$. همانطور که این شکل نشان می‌دهد، عوامل $(2x_1+x_0)^2$ و $(4x_2)^2+2(4x_2)(2x_1+x_0)$ فرزندان چپ و راست ریشه می‌باشند.



شکل ٤-۳: نمایش دیاگرام دودویی تیلور تابع $(4x_2+2x_1+x_0)^2$

٤-۲-۳ عملگرهای دیاگرام دودویی تیلور

در این بخش توضیح می‌دهیم که عملگرهای جبری و بولی چگونه عمل می‌کنند. در آغاز عملگرهای جبری مورد بررسی قرار خواهند گرفت. چون عملگر تفریق با استفاده از جمع و ضرب با ۱- ساخته می‌شود، فقط عملگرهای ضرب و جمع را بررسی خواهیم کرد. سپس نشان خواهیم داد که عملگرهای بولی چگونه با استفاده از عملگرهای جبری ساخته می‌شوند.

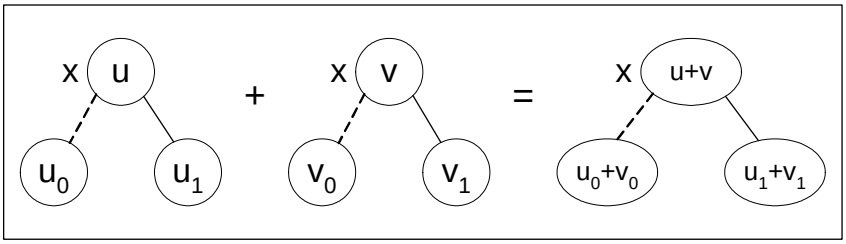
فرض کنید گره‌های u و v باید با هم ترکیب شده و گره q حاصل ترکیب این دو می‌باشد. فرض کنید $\text{var}(u)=x$ و $\text{var}(v)=y$ بیانگر متغیرهای تجزیه مربوط به دو گره می‌باشند.

برای سادگی این الگوریتم‌ها وزن یال‌ها در نظر گرفته نشده‌اند، اما کاملا مشخص است که بزرگترین مقسوم‌علیه مشترک وزن یال‌های خروجی هر گره باید به یال ورودی آن گره منتقل شود.

٤-٢-٣-١ عملگر جمع

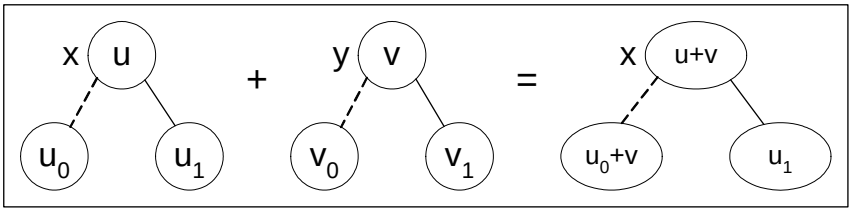
برای جمع دو تابع که بوسیله دیاگرام دودویی تیلور نمایش یافته‌اند، قبل از هر چیز سطح گره‌ها بررسی می‌شود. سطح هر گره فقط به متغیر تجزیه آن گره وابسته است. یک گره در سطح بالاتری از گره دیگر قرار دارد، اگر متغیر تجزیه آن گره بزرگتر از متغیر تجزیه گره دیگر باشد. برای جمع دو گره u و v موارد زیر باید بررسی شود.

١. اگر هر دو گره برچسب‌های یکسانی داشته باشند (شکل ٤-٤)، فرزندان چپ دو گره را با هم جمع کرده و فرزند چپ نتیجه بدست می‌آید. همچنین فرزندان راست دو گره را با هم جمع می‌کنیم تا فرزند راست نتیجه بدست آید.



شکل ٤-٤: جمع دو گره در یک سطح

٢. اگر گره‌ها برچسب‌های مختلفی داشته باشند و $ord(x) > ord(y)$ باشد، آنگاه گره v با فرزند چپ گره u جمع می‌شود.



شکل ٤-٥: جمع دو گره در سطوح مختلف

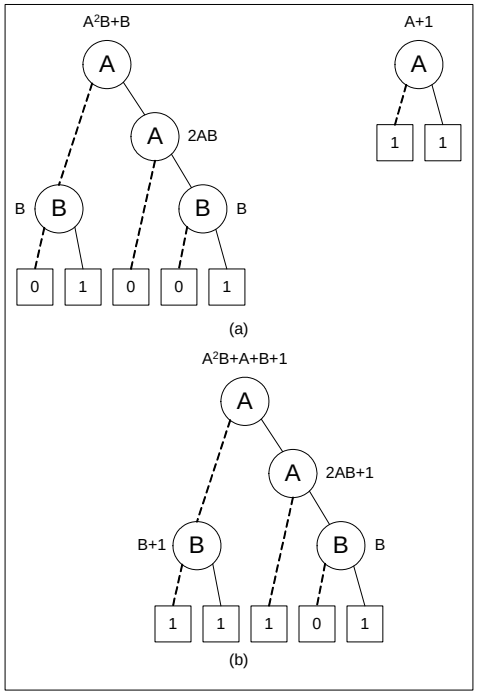
این شیوه به صورت بازگشتی اعمال می‌گردد تا زمانی که به گره ترمینال برسیم. زمانی که یکی از دو گره ترمینال باشد روش مشابه می‌باشد و وقتی هر دو گره ترمینال هستند، نتیجه یک گره ترمینال با مقدار برابر جمع مقادیر دو ترمینال می‌باشد.

شکل ٤-٦ این الگوریتم را روشن‌تر می‌نماید. به دلیل ساختمان‌داده دودویی ، پیاده‌سازی دیاگرام دودویی تیلور ساده‌تر از دیاگرام بسط تیلور می‌باشد [60].

```
function + (btd1, btd2: BTD): BTD;
    result: BTD;
begin
    if btd1 and btd2 are terminal nodes then
        result = btd1.value + btd2.value;
    else
        if btd1 order is greater than btd2 order then
            result.left = btd1.left + btd2;
            result.right = btd1.right;
        else if btd2 order is greater than btd1 order then
            result.left = btd2.left + btd1;
            result.right = btd2.right;
        else // btd1 and btd2 have equal order
            result.left = btd1.left + btd2.left;
            result.right = btd1.right + btd2.right;
        endif
    endif
endfunction
```

شکل ٤-٦: شبه‌کد عمل جمع

مثال۳: شکل ٤-۷(الف) دو معادله جبری A^2B+B و $A+1$ را نشان می‌دهد. حاصل‌جمع این دو در شکل ٤-۷(ب) نشان داده شده است.

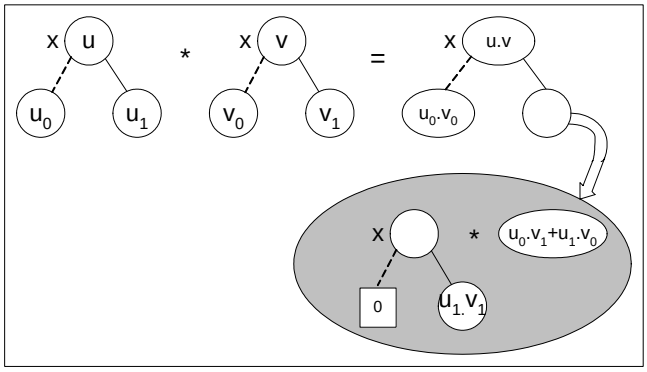


شکل ٤-۷: مثالی از جمع دو دیاگرام دودویی تیلور

٤-۲-۳-۲ عملگر ضرب

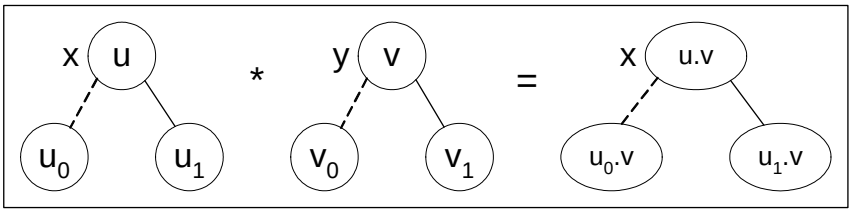
برای ضرب دو گره u و v موارد زیر باید بررسی شود.

۱. اگر برچسب گره‌ها یکسان باشد (شکل ٤-۸)، فرزندان چپ دو گره با هم ضرب شده و حاصل فرزند چپ نتیجه می‌باشد. آنگاه فرزندان راست دو گره با هم ضرب می‌گردند و حاصل، فرزند راست فرزند راست نتیجه می‌باشد (دو سطح پایین‌تر). سپس حاصل‌جمع حاصل‌ضرب متقاطع فرزندان دو گره محاسبه و با فرزند راست نتیجه جمع می‌شود.



شکل ٤-۸: ضرب دو گره در یک سطح

۲. اگر گره‌ها برچسب‌های مختلفی داشته باشند و $ord(x) > ord(y)$ باشد، آنگاه گره v در هر دو فرزند گره u ضرب می‌شود.



شکل ٤-۹: ضرب دو گره در سطوح مختلف

الگوریتم به صورت بازگشتی فراخوانی می‌گردد تا به گره‌های ترمینال برسیم. شکل ٤-۱۰ شبه‌کد این الگوریتم را نشان می‌دهد. همانطور که کاملا مشهود است، عملگر ضرب در دیاگرام دودویی تیلور ساده‌تر از معادل دیاگرام بسط تیلور خود [60] می‌باشد و بنابراین پیاده‌سازی آن ساده‌تر و سریع‌تر خواهد بود.

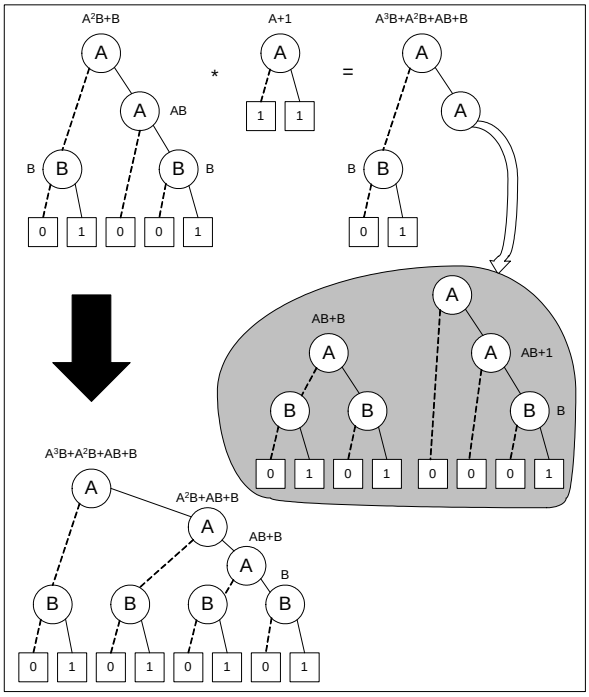
```
function * (btd1, btd2: BTD): BTD;
    result: BTD;
    t1, t2, t3, t4: BTD;
begin
    if btd1 and btd2 are terminal nodes then
        result = btd1.value * btd2.value;
    else
        if btd1 order is greater than btd2 order then
            result.left = btd1.left * btd2;
            result.right = btd1.right * btd2;
        else if btd2 order is greater than btd1 order then
            result.left = btd2.left * btd1;
            result.right = btd2.right * btd1;
        else // btd1 and btd2 have equal order
            t1 = btd1.left * btd2.right;
            t2 = btd1.right * btd2.left;
            t3 = t1 + t2;

            t4.left = BTD_ZERO;
            t4.right = bt1.right * bt2.right;

            result.left = btd1.left * btd2.left;
            result.right = t3 + t4;
        endif
    endif
endfunction
```

شکل ٤-١٠: شبه‌کد عمل ضرب

مثال ٤: شکل ٤-١١ عمل ضرب دو عبارت A^2B+B در $A+1$ را نشان می‌دهد.



شکل ٤-١١: مثالی از ضرب دو دیاگرام دودویی تیلور

٤-۲-۳-۳ عملگرهای بولی

در این بخش نشان می‌دهیم که چگونه توابع بولی بوسیله دیاگرام دودویی تیلور نمایش پیدا می‌کنند. از آنجا که دیاگرام دودویی تیلور یک پیاده‌سازی از دیاگرام بسط تیلور می‌باشد، عملگرهای بولی آن مشابه عملگرهای بولی دیاگرام بسط تیلور می‌باشند. با محدود کردن دامنه توابع به $\{0,1\}$ و تعریف توابع بولی به صورت توابع جبری، می‌توان توابع بولی را توسط دیاگرام دودویی تیلور نمایش داد. عملگرهای بولی NOT، AND، OR و XOR به صورت زیر تعریف می‌شوند.

رابطه ٤-٦
$$NOT(x) = \overline{x} = 1 - x$$

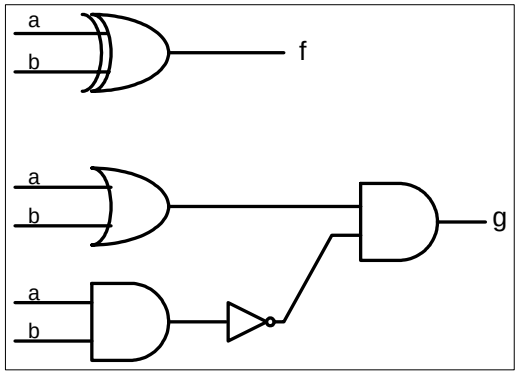
رابطه ٤-٧
$$AND(x, y) = x \times y$$

رابطه ٤-٨
$$OR(x, y) = x + y - x \times y$$

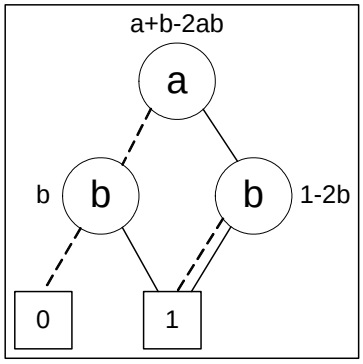
رابطه ٤-٩
$$XOR(x, y) = x + y - 2 \times x \times y$$

مثال ٥: شکل ٤-۱۲ دو مدار بولی را نشان می‌دهد. برای اثبات برابری این دو مدار، ابتدا عملگرهای بولی را به عملگرهای جبری تبدیل می‌کنیم. سپس دیاگرام دودویی تیلور مربوط به این دو طرح ساخته شده وبا هم مقایسه می‌گردند. همانطور که مشخص است دیاگرام دودویی تیلور این دو طرح با هم برابر می‌باشند.

رابطه ٤-۱۰
$$\begin{aligned} f &= Xor(a,b) = a + b - 2 \times a \times b \\ g &= And(Or(a,b), Not(And(a,b))) = \\ &\quad And(a + b - a \times b, 1 - a \times b) = \\ &\quad (a + b - a \times b) \times (1 - a \times b) \end{aligned}$$



شکل ٤-۱۲: مدارهای بولی نمایش دهنده f و g



شکل ٤-١٣: دیاگرام دودویی تیلور توابع f و g

٤-٢-٤ قواعد ساده‌ساز دیاگرام دودویی تیلور

از آنجا که دیاگرام دودویی تیلور یک پیاده‌سازی از دیاگرام بسط تیلور می‌باشد، قواعد ساده‌ساز دیاگرام دودویی تیلور مشابه دیاگرام بسط تیلور می‌باشد. برای رسیدن به ساده‌ترین دیاگرام دودویی تیلور ممکن، باید از قواعد زیر پیروی کرد:

قاعده ١: گره‌های اضافی را حذف می‌کنیم. گره اضافی گره‌ای است که فرزند راستش ترمینال 0 بوده و یا وزن یال ورودی‌اش صفر باشد (فقط ترمینال 0 وزن ورودی صفر دارد). گره‌های اضافی با فرزند چپ خود جایگزین می‌شوند.

قاعده ٢: گره‌های مشابه با هم ادغام[1] می‌شوند. دو گره مشابه هستند اگر برچسب‌های یکسانی داشته باشند و فرزندان آنها هم مشابه باشند. در صورتی که دو گره مشابه وجود داشته باشد یکی از آن دو را حذف کرده و همه یال‌های ورودی گره حذف‌شده را به گره دیگر اشاره می‌دهیم.

٤-٣ خاصیت یگانی دیاگرام دودویی تیلور

در این بخش ثابت خواهیم کرد که دیاگرام دودویی تیلور یگانی می‌باشد (بین توابع جبری و دیاگرام‌های دودویی تیلور یک تناظر یک‌به‌یک یکتا وجود دارد).

وقتی که بین متغیرها ترتیب مشخصی وجود داشته باشد و گره‌های مشابه توسط قواعد ساده‌ساز حذف گردیده باشند و راهکار مشخصی برای نسبت دادن وزن به یال‌ها (بزرگترین مقسوم‌علیه مشترک) وجود داشته باشد، یگانی بودن دیاگرام دودویی تیلور تحقق می‌یابد.

[1] Merge

اثبات:

اثبات این قضیه بدیهی است. نشان خواهیم داد که دیاگرام بسط تیلور یگانی می‌باشد و از روی آن یگانی بودن دیاگرام دودویی تیلور را نتیجه خواهیم گرفت.

در هر مرحله از ساختن دیاگرام بسط تیلور تابع $f(x)$، $f(0)$ و همه مشتقاتش ($f'(0), f''(0), \cdots$) باید محاسبه گردند. همچنین برای دو تابع $f(x)$ و $g(x)$ داریم:

$$g(0)=h(0), g'(0)=h'(0), g''(0)=h''(0), \cdots \Leftrightarrow f(x)=g(x) \qquad \textbf{رابطه ٤-١١}$$

این بدان معنی است که دو تابع مساوی هستند اگر و تنها اگر همه مشتقاتشان با هم مساوی باشند. از این قاعده می‌توان نتیجه گرفت که دیاگرام بسط تیلور دو تابع با هم برابر هستند اگر و تنها اگر دو تابع با هم برابر باشند.

به صورت مشابه، در هر مرحله از ساختن دیاگرام دودویی تیلور نیز مشتقات تابع $f(x)$ محاسبه می‌گردند.

در مرحله ۱، $f(0)$ محاسبه می‌گردند.

در مرحله ۲، $f'(0)$ محاسبه می‌گردند.

در مرحله ۳، $f''(0)/2!$ محاسبه می‌گردند.

⋮

در مرحله n، $f^{(n-1)}(0)/(n-1)!$ و $f^{(n)}(0)/n!$ محاسبه می‌گردند.

همانطور که مشهود است، $f(0)$ و همه مشتقاتش ($f'(0), f''(0), \cdots$) در ساختار دیاگرام دودویی تیلور تابع $f(x)$ به کار رفته و هر کدام جایگاه مشخصی دارند. بنابراین از رابطه ٤-١١ می‌توان نتیجه گرفت که دیاگرام دودویی تیلور یگانی می‌باشد.

٤-٤ نتایج تجربی

کاملا مشخص است که بدلیل ساختمان‌داده ساده‌تر، پیاده‌سازی دیاگرام دودویی تیلور ساده‌تر از دیاگرام بسط تیلور می‌باشد. بسته دیاگرام دودویی تیلور بوسیله Visual C++ نسخه ٦ پیاده‌سازی گردیده است. همه زمان‌ها برحسب ثانیه ارائه شده است.

برای نشان دادن کارایی دیاگرام دودویی تیلور، چندین آزمایش بر روی معادلات استاندارد انجام شده است. این معادلات به دیاگرام دودویی تیلور تبدیل شده و حافظه مصرفی و مدت زمان تبدیل این معادلات اندازه‌گیری گردیده است. جدول ٤-١ تعداد گره‌ها و مدت زمان تبدیل هر معادله را نشان می‌دهد.

همانطور که جدول ٤-١ نشان می‌دهد، از نظر تعداد گره‌ها و مدت زمان تبدیل، دیاگرام دودویی تیلور بسیار خوب عمل می‌کند. پیچیدگی عملیات جمع و ضرب در دیاگرام دودویی تیلور به ترتیب $O(n+m)$ و $O(n\times m)$ می‌باشد که n و m تعداد گره‌های عملوندها هستند. دلیل این امر آن است که در عملیات جمع، هر گره در دیاگرام دودویی تیلور فقط یکبار پیموده می‌شود. همچنین در عملیات ضرب هر گره ممکن است به اندازه تعداد گره‌های عملوند دوم پیموده شود. این پیچیدگی‌ها با نتایج تجربی جدول ٤-١ سازگاری دارد.

جدول ٤-١: نتایج ساختن دیاگرام دودویی تیلور برای معادلات مختلف

	BTD	
Equations	**Nodes**	**Time**
$\sum_{i=1}^{100} x_i$	102	0
$\sum_{i=1}^{1000} x_i$	1002	0
$\sum_{i=1}^{10000} x_i$	10002	0.1
$\sum_{i=1}^{100000} x_i$	100002	1.1
$\sum_{i=1}^{1000000} x_i$	1000002	9.3
$\prod_{i=1}^{100} x_i$	102	0
$\prod_{i=1}^{1000} x_i$	1002	0
$\prod_{i=1}^{10000} x_i$	10002	0.1
$\prod_{i=1}^{100000} x_i$	100002	1.2
$\prod_{i=1}^{1000000} x_i$	1000002	11.2
$\prod_{i=1}^{100}\sum_{j=1}^{100} x_{i,j}^{\ 2}$	19604	0.4
$\prod_{i=1}^{300}\sum_{j=1}^{300} x_{i,j}^{\ 2}$	178804	3.7

$\prod_{i=1}^{500}\sum_{j=1}^{500} x_{i,j}{}^{2}$	498004	10.4
$\prod_{i=1}^{100}\sum_{j=1}^{100} x_{i,j}{}^{3}$	29405	0.6
$\prod_{i=1}^{300}\sum_{j=1}^{300} x_{i,j}{}^{3}$	268205	5.8
$\prod_{i=1}^{500}\sum_{j=1}^{500} x_{i,j}{}^{3}$	747005	18.2
$\prod_{i=1}^{10}\sum_{j=1}^{10} x_{i,j}(i+j)$	812	0
$\prod_{i=1}^{20}\sum_{j=1}^{20} x_{i,j}(i+j)$	7222	0.1
$\prod_{i=1}^{30}\sum_{j=1}^{30} x_{i,j}(i+j)$	25232	0.6
$\prod_{i=1}^{40}\sum_{j=1}^{40} x_{i,j}(i+j)$	60842	1.4
$\prod_{i=1}^{50}\sum_{j=1}^{50} x_{i,j}(i+j)$	120052	2.8
$\prod_{i=1}^{100}\sum_{j=1}^{100} x_{i,j}(i+j)$	980102	23.9
$\prod_{i=1}^{5}(\sum_{j=1}^{5} x_{i,j}{}^{j})^{i}$	678	0
$\prod_{i=1}^{6}(\sum_{j=1}^{6} x_{i,j}{}^{j})^{i}$	4068	0
$\prod_{i=1}^{7}(\sum_{j=1}^{7} x_{i,j}{}^{j})^{i}$	21480	0.3
$\prod_{i=1}^{8}(\sum_{j=1}^{8} x_{i,j}{}^{j})^{i}$	113772	2
$\prod_{i=1}^{9}(\sum_{j=1}^{9} x_{i,j}{}^{j})^{i}$	527097	13.9
$\prod_{i=1}^{10}(\sum_{j=1}^{10} x_{i,j}{}^{j})^{i}$	Unfeasible	Unfeasible

٤-٥ نتیجه‌گیری

در این فصل یک پیاده‌سازی بهینه به نام دیاگرام دودویی تیلور برای دیاگرام بسط تیلور ارائه گردید. برخلاف دیاگرام بسط تیلور که یک گراف غیر دودویی می‌باشد، دیاگرام دودویی تیلور از یک ساختمان‌داده دودویی استفاده می‌کند. این مساله پیاده‌سازی دیاگرام دودویی تیلور را ساده‌تر می‌سازد.

در فصل‌های آتی کلیه الگوریتم‌ها و روش‌ها برای راحتی فهم روی دیاگرام بسط تیلور ارائه می‌گردند. اما همه آن روش‌ها در سطح پیاده‌سازی از دیاگرام دودویی تیلور استفاده می‌کنند. برای تمام نتایج تجربی بدست آمده در این پایان‌نامه در سطح پیاده‌سازی از دیاگرام دودویی تیلور استفاده شده است.

فصل پنجم: دیاگرام بسط تیلور بهبودیافته

ارزیابی‌رسمی سیستم‌های پیچیده دیجیتال نیازمند راهکاری برای نمایش بهینه توابع جبری و بولی می‌باشد. اگرچه می‌توان دیاگرام بسط تیلور را برای نمایش بهینه توابع جبری به کار برد، این دیاگرام در نمایش توابع بولی بهینه نمی‌باشد. در این فصل تغییراتی به دیاگرام بسط تیلور اعمال می‌گردد تا قابلیت نمایش توابع بولی را افزایش دهد. نتایج تجربی تا ۳۰٪ بهبود عملکرد را در بعضی از مثال‌ها نشان می‌دهد.

۵-۱ مقدمه

توابع بولی اغلب بوسیله دیاگرام‌های تصمیم‌گیری نمایش داده می‌شوند. دیاگرام تصمیم‌گیری دودویی مرتب‌شده[1] [13] معمول‌ترین دیاگرام تصمیم‌گیری در کاربردهای خودکارسازی طراحی الکترونیک[2] می‌باشد [15]. علی‌رغم استفاده گسترده، دسته‌هایی از توابع بولی وجود دارند که نمی‌توان آنها را به نحو بهینه‌ای با این دیاگرام نمایش داد [51][31]. برای نمایش این دسته از توابع بولی، دیاگرام‌های تصمیم‌گیری دیگری پیشنهاد و استفاده شده است. به عنوان مثال، دیاگرام تصمیم‌گیری تابعی مرتب‌شده[3] [52][21] برای نمایش بهینه‌تر طرح‌های مبتنی بر XOR، پیشنهاد گردیده است [53]. دیاگرام تصمیم‌گیری دودویی مرتب‌شده، برای نمایش طرح‌های در سطح گیت به صورت موفقیت آمیزی به کار گرفته شده است؛ اما محدودیت‌های فراوانی در نمایش توابع جبری دارد.

برای نمایش توابع جبری، دیاگرام‌های تصمیم‌گیری سطح‌کلمه[4] پیشنهاد شده است. این دیاگرام‌ها از توابع تجزیه‌ای مشابه توابع تجزیه دیاگرام‌های تصمیم‌گیری بولی استفاده می‌کنند (ولی در سطح جبری). نمونه‌هایی از این دیاگرام‌ها عبارتند از دیاگرام تصمیم‌گیری دودویی با ترمینال‌های متعدد[5] [17][10]، دیاگرام تصمیم‌گیری دودویی با یال‌های وزن‌دار[6] [54]، دیاگرام گشتاور دودویی[7] [16]، دیاگرام تصمیم‌گیری ترکیبی[8] [45]، دیاگرام گشتاور دودویی ضربی[9] [16] و دیاگرام گشتاور دودویی ضربی کرانکر[10] [46][47][48]. این دیاگرام‌ها، نمایش‌های مبتنی بر گراف توابعی با دامنه بولی و برد عددی می‌باشند.

[1] Ordered Binary Decision Diagram
[2] Electronic Design Automation
[3] Ordered Functional Decision Diagram
[4] Word-Level Decision Diagram
[5] Multi Terminal Binary Decision Diagram
[6] Edge-Valued Binary Decision Diagram
[7] Binary Moment Diagram
[8] Hybrid Decision Diagram
[9] Multiplicative Binary Moment Diagram
[10] Kronecker Multiplicative Binary Moment Diagram

با افزایش پیچیدگی سیستم‌های دیجیتال، نیاز به سطوح انتزاع بالاتر بیشتر احساس گردید. دیاگرام بسط تیلور [32][33][34] برای برآوردن این نیاز پیشنهاد شده است. این دیاگرام می‌تواند معادلات با درجه بزرگتر از یک را نمایش دهد. این دیاگرام همچنین می‌تواند برای نمایش توابع با دامنه و برد عددی به کار رود.

اگرچه دیاگرام بسط تیلور در نمایش توابع جبری عملکرد خوبی دارد، در نمایش توابع بولی ضعیف می‌باشد. وقتی یک طرح از بخش‌هایی در سطوح بولی و جبری تشکیل یافته باشد، نمایش دیاگرام بسط تیلور آن طرح به حافظه زیادی نیاز پیدا می‌کند. امروزه بسیاری از ریزپردازنده‌ها از یک مسیرداده[1] بزرگ و چندین واحد کنترل تشکیل یافته‌اند. معمولا مسیرداده شامل مدارهای محاسباتی و واحد کنترل شامل سیگنال‌های تک‌بیتی برای کنترل عملیات مسیرداده می‌باشد. واحد کنترل معمولا به صورت معادلات بولی توصیف می‌شود. در چنین مواردی دیاگرام بسط تیلور راه‌حل مناسبی نمی‌باشد؛ چرا که برای نمایش توابع بولی راه‌حل مناسب و بهینه‌ای ارائه نمی‌دهد.

یک راه‌حل، استفاده از دیاگرام‌های تصمیم‌گیری مختلف برای نمایش بخش‌های مختلف یک طرح می‌باشد. این راه‌حل به پیچیده‌تر شدن فرآیند ارزیابی منجر می‌شود. همچنین استفاده از دو یا چند دیاگرام تصمیم‌گیری مختلف، بررسی برابری[2] دو طرح را غیر ممکن می‌سازد چرا که برابری دو طرح بدان معنی نیست که همه بخش‌های آنها برابر باشند.

هدف این فصل بهبود قابلیت دیاگرام بسط تیلور در نمایش توابع بولی می‌باشد. تغییراتی که در این فصل به دیاگرام بسط تیلور اعمال می‌گردد، قابل اعمال به دیاگرام گشتاور دودویی نیز می‌باشد.

این فصل به صورت زیر سازماندهی شده است. در بخش ۲، مروری مختصر بر دیاگرام بسط تیلور خواهیم داشت. در بخش ۳، دیاگرام بسط تیلور بهبودیافته[3] معرفی می‌گردد. در بخش ٤، قواعد مورد نیاز برای یگانی ساختن دیاگرام بسط تیلور بهبودیافته ارائه خواهد شد. در بخش ٥، مثال‌هایی از دیاگرام بسط تیلور بهبودیافته ارائه می‌شود. در بخش ٦ نشان خواهیم داد که دیاگرام بسط تیلور بهبودیافته می‌تواند تا دو برابر بهتر از دیاگرام بسط تیلور معمولی عمل نماید. نتایج تجربی در بخش ۷ ارائه خواهد شد و نتیجه‌گیری، آخرین بخش این فصل می‌باشد.

[1] Datapath
[2] Equivalence Checking
[3] Improved Taylor Expansion Diagram

۵-۲ مروری بر دیاگرام گشتاور دودویی و دیاگرام بسط تیلور

دیاگرام گشتاور دودویی یک نمایش مبتنی بر گراف است که از رابطه ۵-۱ به عنوان قاعده تجزیه استفاده می‌کند [16].

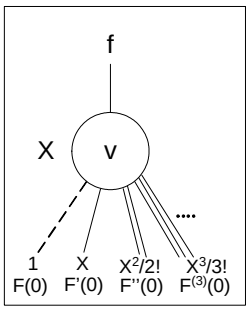
$$f(x) = f(0) + xf_{\alpha x} \qquad \text{رابطه ۵-۱}$$

در رابطه ۵-۱، $f_{\alpha x} = f(1) - f(0)$ بوده و گشتاور خطی f بر حسب x نامیده می‌شود. هر گره در دیاگرام گشتاور دودویی یک برچسب دارد که متغیر مربوط به آن گره را مشخص می‌کند. مشابه بسیاری از دیاگرام‌های تصمیم‌گیری یگانی (مثلا دیاگرام تصمیم‌گیری دودویی)، بین متغیرهای دیاگرام گشتاور دودویی ترتیب مشخصی وجود دارد. هر گره در دیاگرام گشتاور دودویی یک تابع را به صورت تجزیه گشتاوری و بر حسب متغیر آن گره توصیف می‌کند. هر یال در دیاگرام گشتاور دودویی یک وزن ضرب‌شونده دارد که برحسب رابطه ۵-۱ محاسبه می‌شود. درجه خروجی هر گره برابر با ۲ و درجه خروجی هر ترمینال صفر می‌باشد.

دیاگرام بسط تیلور، تعمیم یافته دیاگرام گشتاور دودویی بوده و معادلات با درجه بزرگتر از یک را هم نمایش می‌دهد. این دیاگرام از سری تیلور به عنوان قاعده تجزیه استفاده می‌کند [32][33][34]. سری تیلور تابع مشتق‌پذیر حقیقی $f(x)$ حول $x = 0$ برابر است با

$$f(x) = f(0) + xf'(0) + \frac{1}{2!} f''(0) + \cdots \qquad \text{رابطه ۵-۲}$$

که $f'(0)$ و $f''(0)$ مشتقات اول و دوم تابع f حول $x = 0$ می‌باشند. در دیاگرام بسط تیلور، تابع هر گره به وسیله بسط سری تیلور مطابق رابطه ۵-۲ بدست می‌آید. درجه خروجی هر گره به درجه متغیر وابسته به آن گره بستگی دارد. درجه خروجی یک گره ترمینال برابر با صفر است. کاملا آشکار است که دیاگرام گشتاور دودویی و دیاگرام بسط تیلور در نمایش معادلات درجه اول، بطور مشابه عمل می‌کنند.



شکل ۵-۱: تجزیه در دیاگرام بسط تیلور

شکل ۵-۱ تجزیه دیاگرام بسط تیلور تابع f حول متغیر x را نشان می‌دهد. در این فصل، به مشتق k–ام یک تابع در یک گره، فرزند k–ام آن گره گفته می‌شود: $f(x=0)$ فرزند ۰–ام، $f'(x=0)$ فرزند ۱–ام، $f''(x=0)$ فرزند ۲–ام و .... همچنین به یال‌های متناظر یال–۰ (خط چین)، یال–۱ (خط ممتد) و یال–۲ (خط دوتایی) گفته می‌شود.

دیاگرام گشتاور دودویی و دیاگرام بسط تیلور همچنین توابع جبری و بولی مختلط[1] را هم نمایش می‌دهند. برای نمایش توابع بولی از روابط زیر استفاده می‌شود [16][32][33][34].

رابطه ۵-۳
$$NOT(x) = x' = 1 - x$$

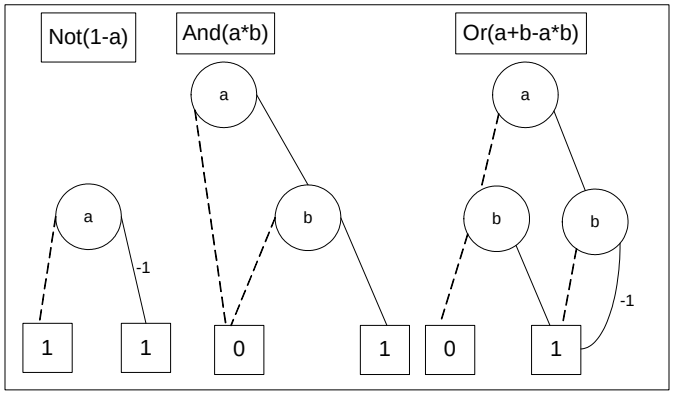
رابطه ۵-٤
$$AND(x,y) = x \wedge y = x * y$$

رابطه ۵-۵
$$OR(x,y) = x \vee y = x + y - x * y$$

از آنجا که دیاگرام بسط تیلور، تعمیم یافته دیاگرام گشتاور دودویی بوده و نمایش بولی هر دو مشابه می‌باشد، در ادامه این فصل فقط از دیاگرام بسط تیلور صحبت خواهد شد. کاملا روشن است که همه تکنیک‌هایی که در ادامه این فصل ارائه خواهد شد قابل اعمال به دیاگرام گشتاور دودویی نیز خواهد بود.

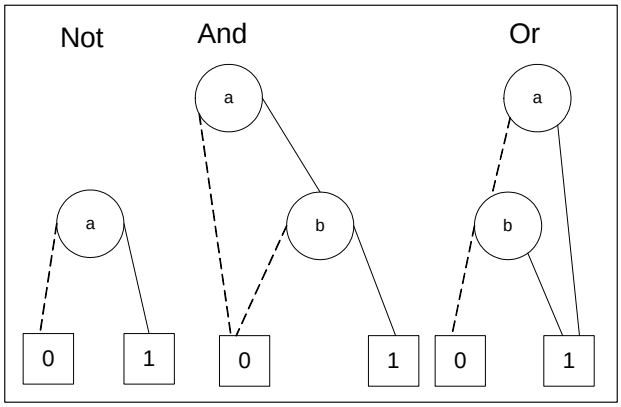
۵-۳ دیاگرام بسط تیلور بهبودیافته

نمایش توابع بولی، یکی از بزرگترین اشکالات دیاگرام بسط تیلور می‌باشد. این امر بدان معنی است که نمایش دیاگرام بسط تیلور بسیاری از توابع بولی، بزرگتر از نمایش دیاگرام تصمیم‌گیری دودویی آنها می‌باشد. برای روشن‌تر شدن قضیه، نمایش دیاگرام بسط تیلور سه تابع اصلی بولی AND، OR و NOT، در شکل ۵-۲ و نمایش دیاگرام تصمیم‌گیری دودویی این توابع در شکل ۵-۳ نشان داده شده است.



شکل ۵-۲: نمایش دیاگرام بسط تیلور توابع اصلی بولی

[1] Mixed



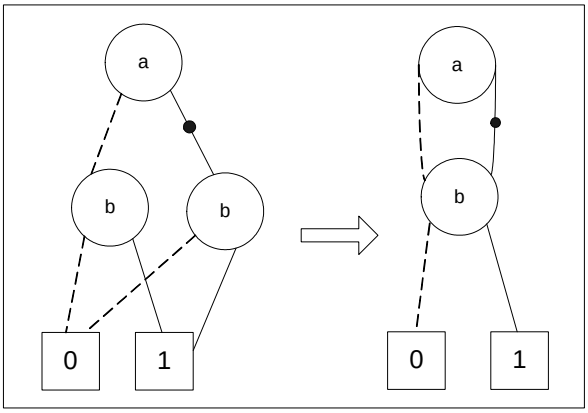
شکل ۵-۳: نمایش دیاگرام تصمیم‌گیری دودویی توابع اصلی بولی

همانطور که مشخص است در مقایسه با دیاگرام تصمیم‌گیری دودویی، در دیاگرام بسط تیلور، توابع AND و NOT با حداقل تعداد گره نمایش پیدا کرده‌اند اما تابع OR تعداد گره‌های بیشتری دارد. از آنجا که تابع OR یکی از توابع پایه‌ای بولی محسوب می‌شود، گره‌های اضافی زیادی هنگام ساختن دیاگرام بسط تیلور ایجاد می‌گردد. بنابراین بهبود نمایش OR در دیاگرام بسط تیلور می‌تواند اندازه نمایش توابع بولی را کاهش دهد.
همانطور که توضیح داده شد، نمایش تابع OR در دیاگرام بسط تیلور از رابطه ۵-۵ بدست می‌آید. اگر متغیر y را به عنوان متغیر وابسته به ریشه در نظر بگیریم، فرزندان ۰ و ۱ به صورت زیر بدست می‌آیند:

- فرزند-۰، که برابر y می‌باشد.
- فرزند-۱، که برابر $y-1$ می‌باشد.

از آنجا که x و y متغیرهای دو مقداری هستند (۰ و ۱)، تابع $y-1$ مکمل منطقی[1] y می‌باشد. از این خاصیت می‌توان برای کوچک کردن گراف استفاده کرد. در حقیقت، زیر گراف نمایش y را می‌توان بین فرزندان ۰ و ۱ ریشه به اشتراک گذاشت. این کار با اضافه کردن یک خصیصه[2] به ساختار یال امکان‌پذیر است. این خصیصه نشان دهنده مکمل تابع گرهی است که یال به آن اشاره می‌کند. برای مثال، در شکل ۵-٤ نمایش جدید دیاگرام بسط تیلور تابع OR نشان داده شده است.

[1] Complement
[2] Attribute



شکل ۵-٤: نمایش دیاگرام بسط تیلور بهبودیافته تابع OR

اگر لازم باشد که یک یال به تابع $f(x)$ اشاره کند، آن یال را به تابع $1-f(x)$ اشاره می‌دهیم و خصیصه آن یال را فعال می‌کنیم. این تغییر باید به گونه‌ای انجام بگیرد که دیاگرام بسط تیلور بهبودیافته یگانی باشد. اگر استفاده از خصیصه یال‌ها را محدود نکنیم، به آسانی می‌توان نشان داد که دیاگرام حاصل یگانی نمی‌باشد. برای مثال رابطه ۵-٦ و رابطه ۵-۷ را در نظر بگیرید.

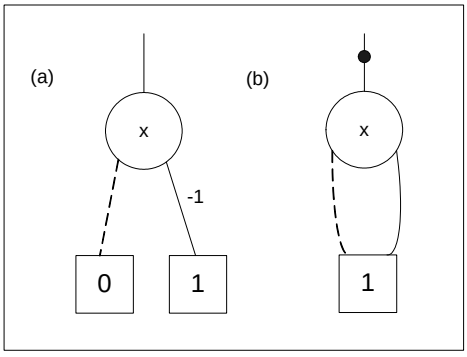
رابطه ۵-٦

$$-x$$

رابطه ۵-۷

$$1-(x+1)$$

آشکار است که رابطه ۵-٦ و رابطه ۵-۷، معادل می‌باشند؛ اما دیاگرام بسط تیلور بهبودیافته این دو رابطه، متفاوت می‌باشد. برای رابطه اول نمایش $-x$ ساخته می‌شود؛ اما برای رابطه دوم نمایش $x+1$ ساخته شده و سپس خصیصه یال اشاره کننده به آن فعال می‌شود. همانطور که در شکل ۵-۵ نشان داده شده است، این دو دیاگرام با هم برابر نمی‌باشند.



شکل ۵-۵: نمایش دیاگرام بسط تیلور بهبودیافته رابطه ۵-٦ و رابطه ۵-۷

در بخش بعدی، قواعدی ارائه می‌گردد که یگانی بودن دیاگرام بسط تیلور بهبودیافته را محقق سازد.

۵-٤ قواعد یگانی

دیاگرام بسط تیلور بهبودیافته یگانی خواهد بود اگر استفاده از خصیصه یال‌ها محدود شود. این محدودیت‌ها در این قسمت ارائه می‌گردند.

تنها در یکی از دو حالت زیر خصیصه یک یال می‌تواند فعال باشد.

۱. بجای ترمینال ۱ از ترمینال ۰ استفاده می‌کنیم (این دو مکمل هستند). در این حالت خصیصه یال اشاره کننده به ترمینال باید فعال باشد.
۲. اگر خصیصه یال-۰ فعال بوده و وزن آن یک باشد، خصیصه یال-۰ را غیر فعال می‌کنیم، وزن سایر یال‌های همسایه را در منهای یک ضرب می‌کنیم و در عوض خصیصه یال ورودی به آن گره را فعال می‌کنیم.

قضیه ۱: قواعد ۱ و ۲ تابع مرتبط با دیاگرام بسط تیلور را تغییر نمی‌دهند.

اثبات: قاعده ۱ می‌گوید که ترمینال ۱ را با ترمینال ۰ جایگزین کرده و خصیصه یالی که به آن اشاره می‌کند را فعال می‌نماییم تا این مساله را نشان دهد. واضح است که این تغییر تابع مرتبط با دیاگرام بسط تیلور را تغییر نمی‌دهد. قاعده ۲ نیاز به دقت بیشتری دارد.

برای ساختن مکمل $f(x)$، تابع $1-f(x)$ باید ساخته شود (رابطه ۵-۳). سری تیلور تابع $1-f(x)$ به صورت زیر می‌باشد.

$$1-f(x)=1-f(0)-xf'(0)-\frac{1}{2!}x^2 f''(0)-\cdots \qquad \text{رابطه ۵-۸}$$

با مقایسه رابطه ۵-۱ و رابطه ۵-۸ مشخص می‌گردد که برای محاسبه مکمل $f(x)$، باید مکمل تابع یال-۰ ($1-f(0)$) محاسبه و وزن سایر یال‌ها در منهای یک ضرب گردد. قاعده ۲ می‌گوید که خصیصه یال‌ها نباید در یال-۰هایی که وزن آنها برابر با یک می‌باشد، فعال باشد؛ چرا که می‌خواهیم خصیصه یال‌ها را تا آنجا که ممکن است بالا ببریم (به سمت ریشه). اگر خصیصه یال-۰ یک گره فعال بوده و وزن آن یک باشد، این گره تابع $1-f(x)$ را نشان می‌دهد. بنابراین با غیر فعال کردن خصیصه یال-۰ و ضرب کردن وزن سایر یال‌ها در منهای یک، تابع گره را مکمل می‌کنیم ($1-(1-f(x))=f(x)$) و در عوض خصیصه یال ورودی به گره را فعال می‌نماییم.

قضیه ۲: دیاگرام بسط تیلور بهبودیافته که با استفاده از قواعد ۱ و ۲ ساخته شده باشد، یگانی می‌باشد.

اثبات: اثبات این قضیه با استقرا[1] روی اندازه مجموعه آرگومان‌های تابع f انجام می‌گیرد.

اگر اندازه مجموعه آرگومان‌ها صفر باشد، تابع f یک تابع ثابت می‌باشد. بنابراین تنها گره این دیاگرام یکی از دو ترمینال ۰ یا ۱ خواهد بود (ترمینال ۱ بوسیله ترمینال ۰ و یال اشاره‌کننده با خصیصه فعال ساخته می‌شود). یال اشاره‌کننده به این گره یک وزن و یک خصیصه دارد. فرض کنید برای یک تابع ثابت، دو دیاگرام بسط تیلور بهبودیافته متفاوت وجود داشته باشد. در دیاگرام بسط تیلور بهبودیافته هنگامی که فقط یک گره وجود دارد (ترمینال ۰)، وزن یال‌ها مشابه وزن یال‌ها در دیاگرام بسط تیلور معمولی می‌باشد. چرا که در این حالت فقط قاعده ۱ می‌تواند اعمال شود و این قاعده وزن یال‌ها را عوض نمی‌کند. بنابراین تفاوت بین این دو در خصیصه یال‌ها و یا در نوع ترمینال‌ها خواهد بود. از آنجا که در دیاگرام بسط تیلور بهبودیافته تنها از ترمینال ۰ استفاده می‌شود، تفاوت بین این دو نمی‌تواند از نوع ترمینال‌هایشان باشد. با استناد به قاعده ۱ می‌توان گفت که اگر در دیاگرام بسط تیلور معمولی این گره ترمینال ۱ باشد، در دیاگرام بسط تیلور بهبودیافته خصیصه یال فعال و در غیر این صورت غیرفعال خواهد بود. بنابراین اگر خصیصه یال‌ها در دو دیاگرام بسط تیلور بهبودیافته متفاوت باشند، می‌توان ادعا کرد که دیاگرام‌های بسط تیلور معمولی آن دو هم متفاوت خواهند بود. از آنجا که دیاگرام بسط تیلور معمولی یگانی می‌باشد، تابع این دو دیاگرام بسط تیلور بهبودیافته متفاوت بوده که متناقض فرض اولیه خواهد بود.

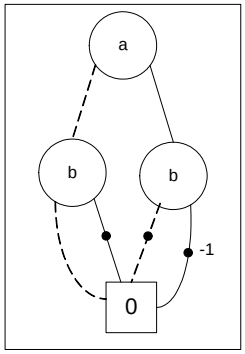
در مرحله بعد فرض می‌کنیم که حکم این قضیه برای توابعی که در مجموعه آرگومان‌های خود کمتر از k متغیر دارند، برقرار باشد ($k \geq 0$). ثابت خواهیم کرد که این قضیه برای توابع k متغیره هم صحیح می‌باشد. فرض کنید که دو دیاگرام بسط تیلور بهبودیافته متفاوت برای چنین تابعی وجود دارد. در این حالت توابع مربوط به فرزندان این دو دیاگرام بسط تیلور بهبودیافته حداکثر $k-1$ متغیره می‌باشند و بنابراین به صورت یگانی نمایش پیدا می‌کنند (قبل از اتصال به ریشه). پس می‌توان ادعا کرد که تفاوت بین این دو دیاگرام بسط تیلور بهبودیافته از یالی که به ریشه اشاره می‌کند یا خود ریشه و یا یالهایی که به فرزندان ریشه اشاره می‌کنند، ناشی شده است. خصیصه یالی‌هایی که به ریشه اشاره می‌کنند، فعال خواهد بود اگر و فقط اگر شرایط قاعده ۲ برقرار باشد. از آنجا که فرزندان ریشه‌ها به صورت یگانی نمایش پیدا کرده‌اند، تفاوت بین این دو دیاگرام بسط تیلور بهبودیافته نمی‌تواند از خصیصه یال‌هایی که به ریشه اشاره می‌کنند، ناشی شده باشد. به صورت مشابه می‌توان نشان داد که تفاوت بین این دو دیاگرام نمی‌تواند از خصیصه یالهایی که به فرزندان ریشه اشاره می‌کنند، ناشی شده باشد.

[1] Induction

بنابراین تفاوت بین این دو از ریشه ناشی شده است. این بدان معنی است که متغیرهای این دو ریشه متفاوت می‌باشند. بنابراین توابع مربوط به این دو دیاگرام بسط تیلور بهبودیافته متفاوت می‌باشند که متناقض فرض اولیه می‌باشد.

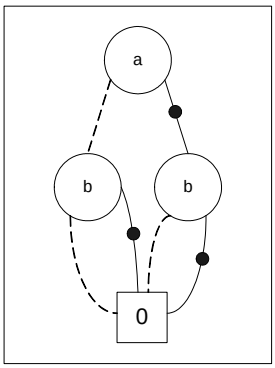
۵-٤-۱ مثال‌هایی از دیاگرام بسط تیلور بهبودیافته

در این بخش کاربرد قواعد بالا از طریق مثال‌های متعدد مشخص می‌گردد. نمایش دیاگرام بسط تیلور تابع OR را در نظر بگیرید (شکل ۵-۲). در اولین قدم ترمینال ۱ را با ترمینال ۰ جایگزین می‌کنیم. بر طبق قاعده ۱ خصیصه یال‌هایی که به ترمینال ۱ اشاره می‌کرده‌اند، را فعال می‌نماییم. نتیجه این مرحله در شکل ۵-٦ نمایش داده شده است.



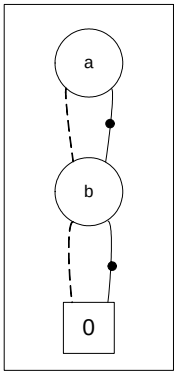
شکل ۵-٦: اعمال قاعده ۱ به دیاگرام بسط تیلور شکل ۵-۲

قاعده ۲ بیان می‌دارد که خصیصه یال‌های ۰ غیر فعال گردد، وزن یال‌های همسایه در منهای یک ضرب شود و خصیصه یال ورودی آن گره فعال گردد. این قاعده به گره‌هایی که یال-۰ با خصیصه فعال و وزن یک دارند اعمال می‌گردد. خروجی این مرحله در شکل ۵-۷ نشان داده شده است.



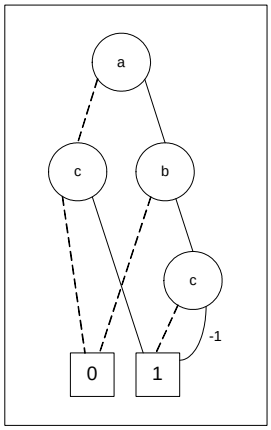
شکل ۵-۷: اعمال قاعده ۲ به دیاگرام بسط تیلور شکل ۵-۲

در آخرین مرحله، گره‌های مشابه ادغام می‌گردند. همانطور که در شکل ۵-۷ نشان داده شده است، گره‌های b مشابه می‌باشند. بنابراین این دو را با هم ادغام می‌کنیم. خروجی این مرحله و دیاگرام بسط تیلور بهبودیافته نهایی در شکل ۵-۸ نشان داده شده است.



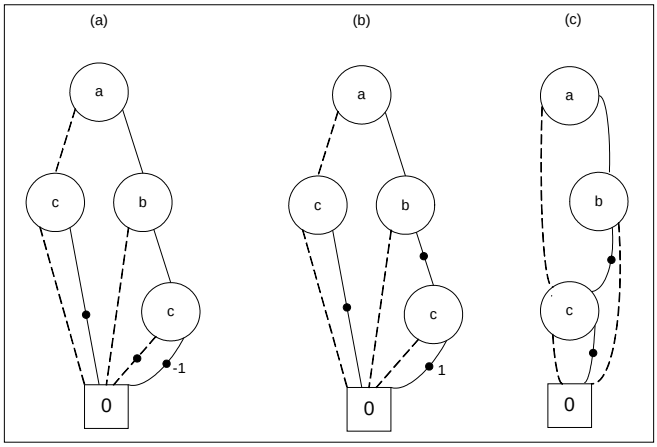
شکل ۵-۸: دیاگرام بسط تیلور بهبودیافته تابع OR

کارایی این شیوه وقتی آشکار می‌گردد که نمایش دیاگرام بسط تیلور بهبودیافته توابع بولی پیچیده‌تر با نمایش دیاگرام بسط تیلور معمولی مقایسه گردد. به عنوان مثال، تابع $(a \wedge b) \vee c$ را در نظر بگیرید. دیاگرام بسط تیلور این تابع در شکل ۵-۹ نشان داده شده است.



شکل ۵-۹: نمایش دیاگرام بسط تیلور تابع $(a \wedge b) \vee c$

این گراف با استفاده از قواعد قبلی ساده می‌شود. شکل ۵-۱۰-الف گراف خروجی را پس از اعمال قاعده ۱ نشان می‌دهد. شکل ۵-۱۰-ب گراف خروجی را پس از اعمال قاعده ۲ نشان می‌دهد. همانطور که مشخص است، تعدادی گره مشابه در گراف تولید شده است. آخرین مرحله، ادغام گره‌های مشابه می‌باشد. شکل ۵-۱۰-پ نمایش نهایی دیاگرام بسط تیلور بهبودیافته را نشان می‌دهد.



شکل ۵-۱۰: مراحل بهبود بخشیدن دیاگرام بسط تیلور تابع $(a \wedge b) \vee c$

۵-۵ بررسی موردی

اصل موضوع ۱: تعداد گره‌های مورد نیاز برای نمایش $f = OR_{i=1}^{n} x_i$ بوسیله دیاگرام بسط تیلور از رابطه بازگشتی زیر بدست می‌آید.

$$TotalNumNodes_{TED}(n) = TotalNumNodes_{TED}(n-1) + 2 \qquad \textbf{رابطه ۵-۹}$$

اثبات: رابطه جبری زیر باید بوسیله دیاگرام بسط تیلور نمایش پیدا کند.

$$f = x_1 + x_2 + \cdots + x_n - x_1x_2 - x_1x_3 - \cdots - x_1x_n - x_2x_3 - \cdots x_{n-1}x_n + x_1x_2x_3 + \cdots - \cdots + x_1x_2x_3 \cdots x_n$$

فرض کنید که متغیر ریشه x_1 باشد(چون رابطه جبری بالا بر حسب همه متغیرها متقارن می‌باشد، ترتیب میان متغیرها تاثیری در اندازه دیاگرام حاصل نخواهد داشت). آنگاه فرزند-۰ ریشه برابر با $f(x_1 = 0)$ و فرزند-۱ ریشه برابر با $\frac{df}{dx_1}(x_1 = 0)$ خواهند گردید.

$$f(x_1 = 0) = x_2 + x_3 + \cdots + x_n - x_2x_3 - x_2x_4 - \cdots - x_2x_n - x_3x_4 - \cdots - x_{n-1}x_n + x_2x_3x_4 + \cdots - \cdots + x_2x_3 \cdots x_n$$

$$\frac{df}{dx1}(x1 = 0) = 1 - x_2 - x_3 - \cdots x_n + x_2x_3 + x_2x_4 + \cdots + x_2x_n + x_3x_4 + \cdots + x_{n-1}x_n - x_2x_3x_4 - \cdots + \cdots - x_2x_3 \cdots x_n$$

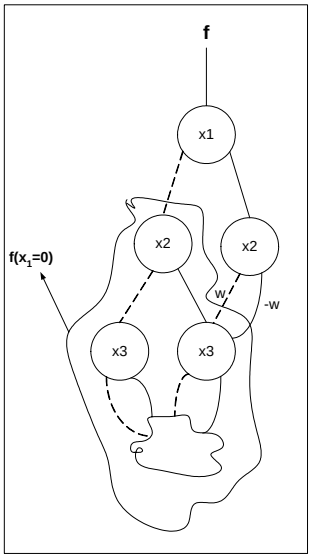
مشخص است که $f(x_1 = 0)$ برابر با $OR_{i=2}^{n} x_i$ می‌باشد. بنابراین به $TotalNumNodes_{TED}(n-1)$ تعداد گره برای ساخته شدن نیاز دارد. از طرف دیگر، $\frac{df}{dx_1}(x_1 = 0)$ برابر با $1 - OR_{i=2}^{n}(x_i) = not(OR_{i=2}^{n}(x_i))$

می‌باشد. فرض کنید که متغیر ریشه این تابع، x_2 باشد. فرزند-۰ گره جدید، برابر با $\frac{df}{dx_1}(x_1=0,x_2=0)$ و فرزند-۱ برابر با $\frac{df}{dx_1dx_2}(x_1=0,x_2=0)$ خواهند گردید.

$$\frac{df}{dx_1}(x_1=0,x_2=0)=1-x_3-\cdots-x_n+x_3x_4+x_3x_5+\cdots+x_3x_n+\cdots+x_{n-1}x_n-x_3x_4x_5-$$
$$\cdots+\cdots-x_3x_4\cdots x_n$$
$$\frac{df}{dx_1dx_2}(x_1=0,x_2=0)=-1+x_3+\cdots+x_n-x_3x_4-x_3x_5-\cdots-x_3x_n-\cdots-x_{n-1}x_n+x_3x_4x_5+$$
$$\cdots-\cdots+x_3x_4\cdots x_n$$

با مقایسه رابطه‌های قبلی مشخص می‌گردد که $\frac{df}{dx_1}(x_1=0,x_2=0)$ برابر با $1-OR_{i=3}^{n}(x_i)=not(OR_{i=3}^{n}(x_i))$ می‌باشد. همچنین، این تابع هنگام نمایش تابع $f(x_1=0)$ ساخته شده است. علاوه بر این، تابع $\frac{df}{dx_1dx_2}(x_1=0,x_2=0)$ منهای یک برابر $\frac{df}{dx_1}(x_1=0,x_2=0)$ می‌باشد. بنابراین با متصل کردن یک یال با وزن منفی به گرهی که تابع $\frac{df}{dx_1}(x_1=0,x_2=0)$ را نشان می‌دهد، قابل نمایش خواهد بود.

با توجه به مطالب فوق، برای نمایش تابع $f=OR_{i=1}^{n}x_i$ بوسیله دیاگرام بسط تیلور به $TotalNumNodes_{TED}(n)=2n-1$ گره نیاز می‌باشد.

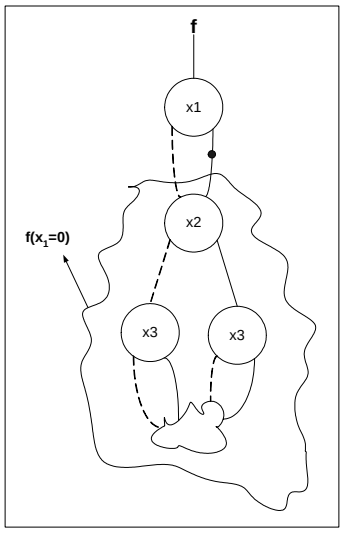


شکل ۵-۱۱: نمایش دیاگرام بسط تیلور تابع $\underset{i=1}{\overset{n}{Or}}\,x_i$

اصل موضوع ۲: تعداد گره‌های مورد نیاز برای نمایش $f = OR_{i=1}^{n} x_i$ بوسیله دیاگرام بسط تیلور بهبودیافته از رابطه بازگشتی زیر بدست می‌آید.

$$TotalNumNodes_{ITED}(n) = TotalNumNodes_{ITED}(n-1) + 1 \qquad \textbf{رابطه ۵-۱۰}$$

اثبات: اثبات اصل موضوع ۲ مشابه اثبات اصل موضوع ۱ می‌باشد. اما برای نمایش فرزند-۱، یال-۱ ریشه با خصیصه فعال، به فرزند-۰ ریشه اشاره می‌کند چرا که فرزندان ۰ و ۱ ریشه مکمل هم می‌باشند.
بنابراین برای نمایش تابع $f = OR_{i=1}^{n} x_i$ بوسیله دیاگرام بسط تیلور بهبودیافته، به $TotalNumNodes_{TED}(n) = n$ گره نیاز می‌باشد.



شکل ۵-۱۲: نمایش دیاگرام بسط تیلور بهبودیافته تابع $\overset{n}{Or}_{i=1} x_i$

۵-۶ نتایج تجربی

در این قسمت چندین طرح مختلف به دیاگرام بسط تیلور معمولی و بهبودیافته تبدیل شده و نتایج حاصل از تبدیل این دو با هم مقایسه شده است. نتایجی که در این قسمت ارائه می‌گردد روی ماشین پنتیوم[1] ٤ با یک گیگابایت[2] حافظه بدست آمده است. همه زمان‌ها برحسب ثانیه ارائه شده است. هر دو بسته دیاگرام بسط تیلور معمولی و بهبودیافته در محیط VisualC++ نسخه ٦ پیاده‌سازی و کامپایل گردیده‌اند. جدول ۵-۱ نتایج بدست

[1] Pentium
[2] Gigabyte

آمده از تبدیل چندین طرح در سطح گیت را نشان می‌دهد. دیاگرام تصمیم‌گیری دودویی همه این طرح‌ها، حداقل ۵۰ گره دارد. ستونی که با BDD مشخص شده است، نتایج تبدیل این طرح‌ها به دیاگرام تصمیم‌گیری دودویی را نشان می‌دهد. همچنین زیر ستون‌های مربوطه، تعداد گره‌های مصرفی و زمان تبدیل را نشان می‌دهند. به طور مشابه ستون‌هایی که با TED و Improved TED مشخص شده‌اند، نتایج تبدیل این طرح‌ها به دیاگرام بسط تیلور و دیاگرام بسط تیلور بهبودیافته را نشان می‌دهند.

جدول ۵-۱: مقایسه چندین ساختمان‌داده مختلف بوسیله مدارهای سطح گیت

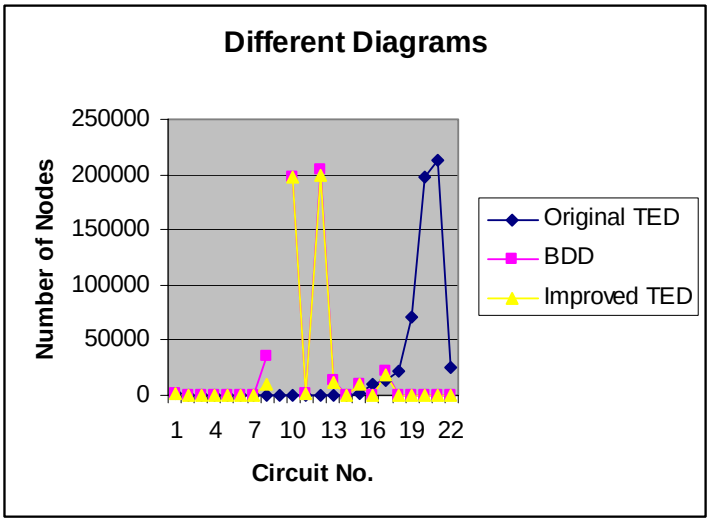
Circuits	Inputs	Outputs	Nets	Gates	TED		Improved TED		BDD	
					Time	Nodes	Time	Nodes	Time	Nodes
Hamming (8 bit)	8	7	94	96	0.3	1851	0.5	1765	0.0	1570
Address Decoder	21	4	23	27	0.0	150	0.0	111	0.0	108
Parity Gen. (15 bit)	15	2	45	47	0.0	423	0.1	415	0.0	401
Parity Gen. (11 bit)	11	2	33	35	0.0	391	0.0	367	0.0	364
Parity Gen. (27 bit)	27	2	86	91	0.1	656	0.3	623	0.0	603
Parity Gen. (36 bit)	36	2	99	102	0.6	823	0.9	802	0.1	796
Array Divider (8 bit)	16	16	476	485	0.0	375	0.0	261	0.0	170
Array Divider Multiplier (8 bit)	17	16	1399	1412	1.1	71464	1.8	34902	0.8	10234
Comparator (16 bit)	32	1	78	79	8.0	197921	9.8	197921	1.7	197889
Comparator (8 bit)	16	1	38	39	0.0	963	0.0	955	0.0	955
Simple Adder (16 bit)	16	8	123	142	10.0	213543	11.1	203567	7.8	199344
CLA (8 bit)	17	9	86	95	16.3	12895	23.3	12885	0.1	11466
CLA (4 bit)	9	5	42	47	0.1	693	0.1	687	0.0	612
FAdder (8 bit)	17	8	84	92	15.2	10358	21.8	10348	0.1	9437
FAdder (4 bit)	9	4	40	44	0.0	548	0.0	542	0.0	495
FAdder (16 bit)	33	16	4166	186	4.1	22345	7.2	22213	3.4	18324
CSA (8 bit)	25	16	80	96	0.0	408	0.0	384	0.0	352
CSA (16 bit)	49	32	160	192	0.1	808	0.2	768	0.0	704
CPA (6 bit)	13	9	54	62	0.0	473	0.0	465	0.0	445
CPA (8 bit)	17	11	78	93	0.0	567	0.0	544	0.0	532
Mux (2 * 4)	12	4	18	22	0.0	85	0.0	77	0.0	64
Total					**55.9**	**537740**	**77.1**	**490602**	**14**	**454865**
Average					**2.661905**	**25606.667**	**3.671429**	**23362**	**0.666667**	**21660.24**

این جدول همچنین تعداد ورودی‌ها، خروجی‌ها و گیت‌ها را برای هر طرح نشان می‌دهد. همانطور که مشاهده می‌شود، تعداد گره‌ها در دیاگرام بسط تیلور بهبودیافته همیشه کمتر از دیاگرام بسط تیلور می‌باشد. اما دیاگرام بسط تیلور بهبودیافته از نظر زمانی ضعیف‌تر عمل می‌کند؛ زیرا پیچیدگی الگوریتم در این دیاگرام بیشتر از دیاگرام بسط تیلور می‌باشد. لازم به ذکر است که ترتیب متغیرها در هر سه دیاگرام یکسان می‌باشد.

دیاگرام بسط تیلور بهبودیافته هنگامی بهتر عمل می‌کند که تفاوت بین تعداد گره‌های دیاگرام تصمیم‌گیری دودویی و دیاگرام بسط تیلور معمولی قابل توجه باشد؛ چرا که در این مواقع دیاگرام بسط تیلور بهبودیافته امکان بهینه‌سازی بیشتری دارد.

طرح‌هایی که برای این آزمایش انتخاب شده‌اند، واحدهای محاسباتی واقعی بوده و در ساخت واحدهای بزرگتر استفاده می‌شوند. جدول ۵-۱ نشان می‌دهد که دیاگرام بسط تیلور بهبودیافته به طور متوسط ۹٪ بهتر از دیاگرام بسط تیلور معمولی عمل کرده است. در عین حال این دیاگرام به طور متوسط ۷٪ ضعیف‌تر از دیاگرام تصمیم‌گیری دودویی می‌باشد.

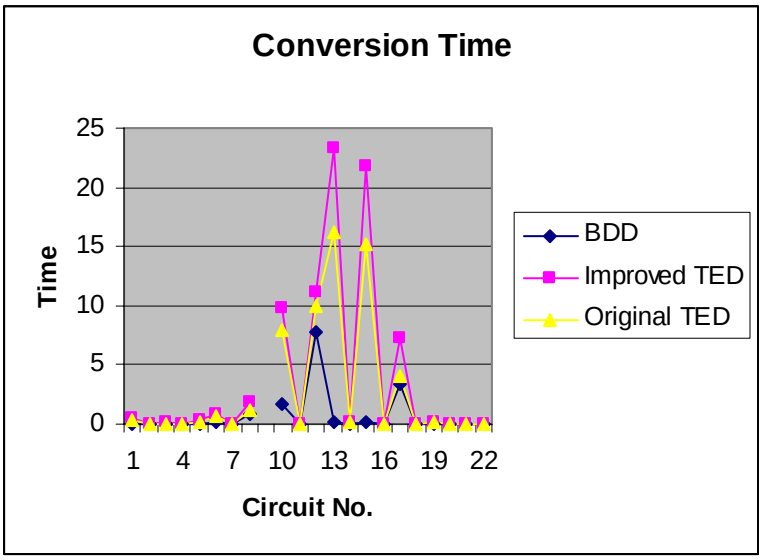
شکل ۵-۱۳ نمودار[1] تعداد گره‌ها در سه دیاگرام را نشان می‌دهد. همانطور که این نمودار نشان می‌دهد تعداد گره‌های دیاگرام بسط تیلور بهبودیافته و دیاگرام تصمیم‌گیری دودویی تقریبا یکسان هستند.



شکل ۵-۱۳: تعداد گره‌ها در ساختارهای مختلف

[1] Chart

شکل ۵-۱٤ نمودار زمان تبدیل در سه دیاگرام را نشان می‌دهد. همانطور که مشاهده می‌شود، زمان تبدیل در دیاگرام بسط تیلور بهبودیافته بیشتر از دو دیاگرام دیگر می‌باشد.



شکل ۵-۱٤: زمان تبدیل در ساختارهای مختلف

برای پیاده‌سازی خصیصه یال‌ها از میدان‌های بیتی[1] استفاده شده است. با استفاده از این تکنیک یک بیت از یک متغیر صحیح بزرگ برای خصیصه یال‌ها استفاده شده و مابقی بیت‌ها برای وزن یال‌ها استفاده می‌شوند. بنابراین در مقایسه با دیاگرام بسط تیلور، در دیاگرام بسط تیلور بهبودیافته هر یال تنها یک بیت اضافی دارد. به عنوان مثال، برای یک دیاگرام با ۱۰۰۰۰۰ یال، حافظه اضافی ۲/۵ کیلوبایت می‌گردد که قابل صرفنظر کردن می‌باشد.

```
struct Edge is
{
        signed long int weight : 31;
        unsigned long int attrib : 1;
}
```

شکل ۵-۱۵: شبه‌کد یال در دیاگرام بسط تیلور بهبودیافته

[1] Bit Fields

در دیاگرام بسط تیلور، عملیات بولی به عملیات جبری تبدیل می‌شوند، بنابراین با بهبود بخشیدن عملیات بولی، نمایش جبری هم بهبود پیدا می‌کند. اما از آنجا که عبارات جبری بسیار کوچکتر از عبارات بولی می‌باشند، مقدار بهینه‌سازی محدود می‌باشد. از طرف دیگر نمایش جبری دیاگرام بسط تیلور خوب می‌باشد و به بهبود زیادی نیاز نمی‌باشد. جدول ۵-۲ نتایج تبدیل چند عبارت جبری به دیاگرام بسط تیلور معمولی و بهبودیافته را نشان می‌دهد.

جدول ۵-۲: مقایسه دیاگرام بسط تیلور معمولی و بهبودیافته بوسیله معدلات جبری

Equations	TED		Improved TED	
	Nodes	Time	Nodes	Time
$\sum_{i=1}^{100} x_i$	102	0	101	0
$\sum_{i=1}^{1000} x_i$	1002	0	1001	0
$\sum_{i=1}^{10000} x_i$	10002	0.3	10001	0.4
$\sum_{i=1}^{100000} x_i$	100002	3.8	100001	5.3
$\prod_{i=1}^{100} x_i$	102	0	101	0
$\prod_{i=1}^{1000} x_i$	1002	0	1001	0
$\prod_{i=1}^{10000} x_i$	10002	0.4	10001	0.6
$\prod_{i=1}^{100000} x_i$	100002	4.7	100001	6.4
$\sum_{i=1}^{100} x_i + y_i - x_i y_i$	302	0	300	0
$\sum_{i=1}^{1000} x_i + y_i - x_i y_i$	3002	0.1	3000	0.2
$\prod_{i=1}^{100} x_i + y_i - x_i y_i$	302	0	300	0
$\prod_{i=1}^{1000} x_i + y_i - x_i y_i$	3002	0.3	3000	0.6

۵-۷ نتیجه‌گیری

در این فصل دیاگرام بسط تیلور بهبودیافته پیشنهاد و معرفی شد؛ که در آن، یک خصیصه به یال‌های دیاگرام بسط تیلور اضافه گردید. اگر لازم باشد که یک یال به تابع $1-f(x)$ اشاره کند، آن یال را به تابع $f(x)$ اشاره می‌دهیم و خصیصه آن یال را فعال می‌کنیم. اگرچه اضافه کردن خصیصه به یال‌ها ایده جدیدی نمی‌باشد [55][12]، اما اولین بار است که از آنها برای بهبود نمایش منطقی دیاگرام بسط تیلور استفاده شده است. نتایج تجربی روی محک‌های مختلف کارایی این روش را برای بهبود نمایش منطقی دیاگرام بسط تیلور نشان می‌دهد. از طرف دیگر نمایش جبری دیاگرام بسط تیلور بهبودیافته همچنان خوب است. بنابراین دیاگرام بسط تیلور بهبودیافته برای نمایش طرح‌هایی که از بخش‌های سطح گیت و سطح انتقال‌ثبات تشکیل یافته‌اند، راه‌حل مناسبی می‌باشد.

فصل ششم: دیاگرام بسط تیلور مثبت

ارزیابی‌رسمی سیستم‌های پیچیده دیجیتال نیازمند راهکاری برای نمایش بهینه توابع جبری و بولی می‌باشد. اخیرا یک نمایش یگانی و مبتنی بر گراف به نام دیاگرام بسط تیلور پیشنهاد شده است. اگرچه می‌توان دیاگرام بسط تیلور را برای نمایش بهینه توابع جبری در سطح‌کلمه به کار گرفت، این دیاگرام در نمایش توابع بولی ضعیف می‌باشد. در این فصل تغییراتی به دیاگرام بسط تیلور اعمال می‌گردد که نمایش منطقی در سطح بیت را بهبود بخشد. نشان داده خواهد شد که در نمایش توابع بولی، دیاگرام بسط تیلور تغییریافته مشابه دیاگرام تصمیم‌گیری دودویی عمل می‌کند.

٦-١ مقدمه

توابع بولی اغلب بوسیله دیاگرام تصمیم‌گیری دودویی نمایش می‌یابند. دیاگرام تصمیم‌گیری دودویی مرتب‌شده[1] [13] رایج‌ترین دیاگرام تصمیم‌گیری در کاربردهای خودکارسازی طراحی الکترونیک[2] می‌باشد [15]. دیاگرام تصمیم‌گیری دودویی مرتب‌شده و مشتقاتش به صورت موفقیت‌آمیزی در نمایش طرح‌های سطح گیت به کار گرفته شده‌اند اما در نمایش توابع محاسباتی محدودیت‌هایی دارند.

برای نمایش مدارهای محاسباتی دیاگرام‌های تصمیم‌گیری سطح‌کلمه[3] پیشنهاد شده‌اند. دیاگرام تصمیم‌گیری دودویی با ترمینال‌های متعدد[4] [17][10]، دیاگرام تصمیم‌گیری دودویی با یال‌های وزن‌دار[5] [54]، دیاگرام گشتاور دودویی[6] [16]، دیاگرام تصمیم‌گیری ترکیبی[7] [45]، دیاگرام گشتاور دودویی ضربی[8] [16]، دیاگرام گشتاور دودویی ضربی کرانکر[9] [46][47][48] نمونه‌هایی از دیاگرام‌های تصمیم‌گیری سطح‌کلمه می‌باشند. این دیاگرام‌ها، نمایش‌های مبتنی بر گراف توابعی با دامنه بولی و برد عددی می‌باشند، بنابراین یک تابع جبری باید به معادل بیتی خود تبدیل شود تا توسط دیاگرام‌های تصمیم‌گیری سطح‌کلمه نمایش پیدا کند.

با افزایش پیچیدگی سیستم‌های دیجیتال، نیاز به سطوح انتزاع[10] بالاتر بیشتر احساس گردید. دیاگرام بسط تیلور [32][33][34] به عنوان پاسخی بر این نیاز پیشنهاد شد. این دیاگرام را می‌توان برای نمایش توابع با دامنه جبری

[1] Ordered Binary Decision Diagram
[2] Electronic Design Automation
[3] Word-Level Decision Diagram
[4] Multi Terminal Binary Decision Diagram
[5] Edge-Valued Binary Decision Diagram
[6] Binary Moment Diagram
[7] Hybrid Decision Diagram
[8] Multiplicative Binary Moment Diagram
[9] Kronecker Multiplicative Binary Moment Diagram
[10] Abstraction Levels

و برد جبری به کار برد. بنابراین برخلاف دیاگرام‌های تصمیم‌گیری سطح‌کلمه، یک تابع محاسباتی نباید به معادل بیتی خود تبدیل شود تا توسط دیاگرام بسط تیلور نمایش یابد.

اگرچه دیاگرام بسط تیلور مزایای بسیاری نسبت به دیاگرام‌های تصمیم‌گیری سطح‌کلمه دارا می‌باشد، نمایش ضعیف توابع بولی از مهمترین اشکالات این دیاگرام می‌باشد. بسیاری از طرح‌های پیچیده در سطح انتقال‌ثبات (برای نمونه ریزپردازنده‌ها) شامل یکسری عملگرهای محاسباتی (مثل جمع و ضرب) و چندین بخش بزرگ بولی (مثل واحد کنترل) می‌باشند. در چنین مواردی، دیاگرام تصمیم‌گیری دودویی و مشتقاتش راه‌حل مناسبی نمی‌باشند چرا که در نمایش توابع محاسباتی بهینه نمی‌باشند. همچنین دیاگرام‌های تصمیم‌گیری سطح‌کلمه هم راه‌حل مناسبی نمی‌باشند؛ چرا که توابع محاسباتی را به معادل بیتی خود تبدیل می‌کنند. این نمایش‌ها کارایی ارزیابی‌رسمی در سطح انتقال‌ثبات را کاهش می‌دهند. از طرف دیگر، دیاگرام بسط تیلور که نمایش جبری خوبی دارد راه‌حل مناسبی برای نمایش بولی ارائه نمی‌کند.

یک راه‌حل استفاده از دیاگرام‌های تصمیم‌گیری مختلف برای نمایش بخش‌های مختلف یک طرح می‌باشد. این راه‌حل به پیچیدگی بیشتر در فرآیند ارزیابی‌رسمی منجر می‌شود. همچنین داشتن دو یا چند دیاگرام تصمیم‌گیری، بررسی برابری[1] دو طرح را سخت و حتی غیرممکن می‌سازد چرا که برابری دو طرح به معنای برابری تک تک بخش‌های آن دو نمی‌باشد.

در این فصل یک ساختمان‌داده یکتا برای نمایش عبارات جبری معمول در مسیرداده[2] پردازنده‌ها ارائه می‌گردد. نشان داده خواهد شد که این ساختمان‌داده نمایش بولی خوبی هم دارد.

این فصل به صورت زیر سازماندهی شده است: در بخش ۲، مروری مختصر بر دیاگرام بسط تیلور صورت خواهد گرفت. در بخش ۳ دیاگرام بسط تیلور مثبت ارائه می‌گردد. در بخش ٤ با استفاده از وزن‌های جمع‌شونده بهبود بیشتری به دست خواهد آمد. در بخش ٥ ارتباط بین دیاگرام بسط تیلور مثبت و سایر دیاگرام‌ها بررسی خواهد شد. نتایج تجربی در بخش ٦ ارائه می‌گردد و نتیجه‌گیری در آخرین بخش خواهد آمد.

٦-٢ دیاگرام بسط تیلور

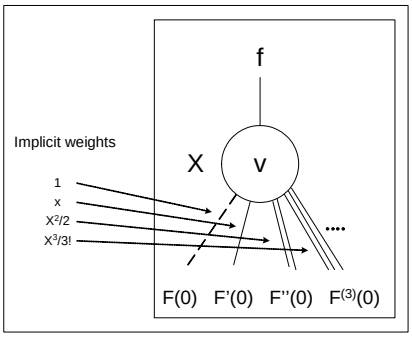
دیاگرام بسط تیلور یک نمایش مبتنی بر گراف است که از سری تیلور به عنوان تابع تجزیه استفاده می‌کند [32][33][34]. بسط تیلور تابع مشتق‌پذیر f حول $x = 0$ برابر است با

[1] Equivalence Checking
[2] Datapath

$$f(x) = f(0) + xf'(0) + x^2/2!\, f''(0) + x^3/3!\, f'''(0) + \cdots$$ **رابطه ٦-١**

که $f'(0)$، $f''(0)$ و $f'''(0)$ مشتق اول، دوم و سوم تابع f حول $x=0$ می‌باشند. تجزیه تابع به صورت بازگشتی از رابطه ٦-١ ادامه می‌یابد. هر گره در دیاگرام بسط تیلور یک برچسب[1] دارد که متغیر وابسته به آنرا مشخص می‌کند. مثل همه دیاگرام‌های تصمیم‌گیری یگانی، متغیرهای دیاگرام بسط تیلور، مرتب‌شده هستند. تابع یک گره بوسیله بسط سری تیلور مطابق رابطه ٦-١ به‌دست می‌آید. درجه خروجی یک گره به درجه متغیر وابسته به آن گره بستگی دارد. درجه خروجی یک گره ترمینال، صفر می‌باشد.

شکل ٦-١ تجزیه دیاگرام بسط تیلور تابع f حول متغیر x را نشان می‌دهد. در این فصل، به مشتق k–ام تابع در یک گره، فرزند–k آن گره گفته می‌شود: $f(x=0)$ فرزند–۰، $f'(x=0)$ فرزند–۱، $f''(x=0)$ فرزند–۲ و ... می‌باشند. همچنین به یال‌های متناظر یال–۰ (خط چین)، یال–۱ (خط ممتد) و یال–۲ (خط دوگانه) گفته می‌شود. هر یال در دیاگرام بسط تیلور یک وزن ضرب‌شونده دارد که از روی سری تیلور به‌دست می‌آید. به اثبات رسیده است که با اعمال محدودیت‌هایی روی وزن و ترتیب متغیرها، دیاگرام بسط تیلور یگانی خواهد بود.



شکل ٦-١: تجزیه در دیاگرام بسط تیلور

برای توابعی که به طور معمول در مشخصه‌های[2] انتقال‌ثبات به کار می‌روند (x^k، $x+y$، $x-y$، $x+y$) دیاگرام بسط تیلور بر حسب تعداد متغیرها خطی می‌باشد. دیاگرام بسط تیلور همچنین می‌تواند توابع بولی را هم نمایش دهد. برای این منظور از روابط زیر استفاده می‌شود [32][33][34].

$$NOT(x) = \bar{x} = 1 - x$$ **رابطه ٦-٢**

[1] Label
[2] Specification

$$AND(x,y) = x \wedge y = x * y$$ **رابطه ٦-٣**

$$OR(x,y) = x \vee y = x + y - x * y$$ **رابطه ٦-٤**

٦-٣ دیاگرام بسط تیلور مثبت

نمایش ضعیف توابع بولی بزرگترین اشکال دیاگرام بسط تیلور می‌باشد. این بدان معنی است که معمولا نمایش دیاگرام بسط تیلور یک تابع بولی از نمایش دیاگرام تصمیم‌گیری دودویی همان تابع بزرگتر است.
دیاگرام بسط تیلور مثبت بر این پایه استوار است که نمایش دیاگرام بسط تیلور وقتی که متغیر تجزیه از درجه یک باشد، می‌تواند ساده شود. این امر می‌تواند نمایش توابع بولی را بسیار ساده سازد. همانطور که در بخش قبلی توضیح داده شد، در هر سطح، دیاگرام بسط تیلور یک تابع را به سری تیلور آن تابع تجزیه می‌کند (رابطه ٦-١). وقتی که متغیر تجزیه از درجه یک باشد، سری تیلور برابر خواهد بود با

$$f(x) = f(0) + xf'(x)$$ **رابطه ٦-٥**

که رابطه زیر در آن صدق می‌کند.

$$f'(0) = f(1) - f(0)$$ **رابطه ٦-٦**

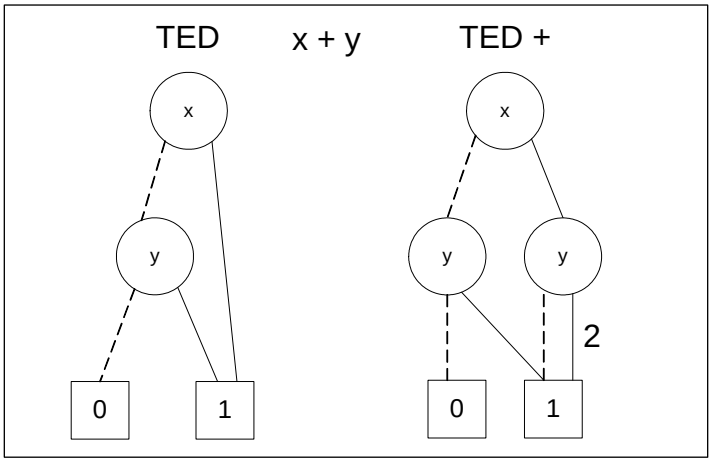
با استفاده از رابطه ٦-٦ در رابطه ٦-٥ و جابجا کردن متغیرها، رابطه ٦-٧ نتیجه می‌شود.

$$f(x) = (1-x) f(0) + xf(1)$$ **رابطه ٦-٧**

رابطه ٦-٧ شبیه تجزیه شانون[1] می‌باشد که به عنوان تابع تجزیه در دیاگرام تصمیم‌گیری دودویی استفاده می‌شود. برای بهبود نمایش توابع بولی در گره‌هایی با دو یال خروجی (گره‌های با درجه یک)، از رابطه ٦-٧ بجای رابطه ٦-٥ به عنوان تابع تجزیه استفاده می‌شود. با انجام این کار و استفاده از رابطه ٦-٢، رابطه ٦-٣ و رابطه ٦-٤ برای نمایش NOT، AND و OR، دیاگرام بسط تیلور مثبت در نمایش توابع بولی مشابه دیاگرام تصمیم‌گیری دودویی عمل می‌کند. از آنجا که به صورت سراسری[2] در همه گره‌ها با دو یال خروجی از رابطه ٦-٧ استفاده می‌شود، دیاگرام بسط تیلور مثبت یگانی می‌باشد.

[1] Shannon Decomposition
[2] Global

اکنون که مزیت دیاگرام بسط تیلور مثبت نسبت به دیاگرام بسط تیلور در نمایش توابع بولی آشکار گردید، مناسب است توجه خود را به توابع جبری معطوف کنیم. آشکار است که دیاگرام بسط تیلور مثبت در نمایش مجموعه‌ای از عملگرهای ضرب مشابه دیاگرام بسط تیلور می‌باشد. اما نمایش عملگر جمع در دیاگرام بسط تیلور مثبت ضعیف‌تر می‌باشد. نمایش دیاگرام بسط تیلور معمولی و مثبت تابع $x+y$ که ضعف دیاگرام بسط تیلور مثبت را نشان می‌دهد، در شکل ٦-۲ نشان داده شده است.



شکل ٦-۲: نمایش دیاگرام بسط تیلور و دیاگرام بسط تیلور مثبت تابع $x+y$

آشکار است که دیاگرام بسط تیلور مثبت در نمایش عمل جمع به خوبی دیاگرام بسط تیلور عمل نمی‌کند. در بخش بعدی، دیاگرام بسط تیلور مثبت برای نمایش بهینه توابع جبری بهبود پیدا خواهد کرد. نشان داده خواهد شد که این بهبود برای نمایش توابع بولی هم مفید می‌باشد.

٦-٤ رسیدن به بهبود بیشتر

در بخش قبلی نشان داده شد که دیاگرام بسط تیلور مثبت در نمایش معادلات جبری به خوبی دیاگرام بسط تیلور عمل نمی‌کند. از طرف دیگر، دیاگرام بسط تیلور مثبت در نمایش عبارات بولی به خوبی دیاگرام تصمیم‌گیری دودویی عمل می‌کند. دیاگرام تصمیم‌گیری دودویی که خصیصه‌ای[1] به یال‌هایش اضافه شده است [55][12] از نظر حافظه بهینه‌تر از دیاگرام تصمیم‌گیری دودویی معمولی عمل می‌کند. با اضافه کردن یال‌های

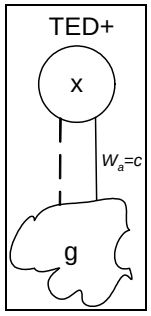
[1] Attribute

وزن‌دار جمع‌شونده به دیاگرام بسط تیلور مثبت می‌توان به چنین کارایی دست یافت. در این بخش با اضافه‌کردن وزن‌های جمع‌شونده، قابلیت دیاگرام بسط تیلور مثبت در نمایش جبری و بولی بهبود پیدا خواهد کرد.

مشخص است که فرزند-۰ تجزیه‌های تیلور و شانون به یک شیوه محاسبه می‌شوند (فرزند-۰ یک گره با قرار دادن ۰ بجای متغیر وابسته به آن گره محاسبه می‌شود). از طرف دیگر در گره‌هایی که بیش از دو یال خروجی دارند، دیاگرام بسط تیلور مثبت از تجزیه تیلور استفاده می‌کند که مشابه دیاگرام بسط تیلور معمولی می‌باشد. می‌توان نتیجه گرفت که ضعف نمایش جبری دیاگرام بسط تیلور مثبت که در بخش قبلی به آن اشاره شد، از فرزند-۱ گره‌هایی با دو یال خروجی ناشی می‌شود.

حالت خاصی از این مورد را با اضافه کردن وزن‌های جمع‌شونده می‌توان برطرف کرد. برای نشان دادن این حالت خاص، گره x تابع $f = g + cx$ را که g یک تابع و c یک ثابت است، در نظر بگیرید. اگر در این گره از تجزیه شانون استفاده شود، فرزند-۱ به شکل $f(x=1) = c + g$ بدست می‌آید. از طرف دیگر اگر از تجزیه تیلور استفاده شود، فرزند-۱ برابر با $f'(x=0) = c$ خواهد بود. با مقایسه این دو آشکار خواهد شد که فرزند-۱ دیاگرام بسط تیلور (c) ساده‌تر از معادل دیاگرام بسط تیلور مثبت خود ($c+g$) می‌باشد. به علاوه، فرزند-۱ دیاگرام بسط تیلور را می‌توان با یک یال که وزن ضرب‌شونده c دارد و به ترمینال ۱ اشاره می‌کند، ساخت.

برای افزایش کارایی دیاگرام بسط تیلور مثبت، وزن‌های جمع‌شونده (علاوه بر وزن‌های ضرب‌شونده) به یال‌ها اضافه شده است. ایده اصلی آن است که تابع g هم‌اکنون برای نمایش فرزند-۰ ($f(x=0) = g$) دیاگرام بسط تیلور مثبت ساخته شده است. بنابراین با اضافه‌کردن وزن‌های جمع‌شونده به یال‌ها، می‌توان فرزند-۱ دیاگرام بسط تیلور مثبت را با یک یال که وزن جمع‌شونده c دارد و به فرزند-۰ ($f(x=0) = g$) اشاره می‌کند، ساخت. شکل ٦-۳ این راهکار را نشان می‌دهد.



شکل ٦-۳: نمایش دیاگرام بسط تیلور مثبت تابع $g + cx$

با اضافه‌کردن وزن‌های جمع‌شونده، تابع یک یال از رابطه $W_a + W_m \times g(x)$ به‌دست می‌آید که W_a وزن جمع‌شونده و W_m وزن ضرب‌شونده و $g(x)$ تابع گرهی است که این یال به آن اشاره می‌کند.
در این بحث ضریب x ثابت بود. اگر ضریب متغیر وابسته به گره ثابت نباشد، ممکن است که نمایش دیاگرام بسط تیلور آن تابع فشرده‌تر از نمایش دیاگرام بسط تیلور مثبت باشد.

٦-٥ نمایش یگانی

برای آنکه دیاگرام بسط تیلور مثبت یگانی گردد، باید محدودیت‌هایی روی وزن یال‌ها اعمال گردد. این محدودیت‌ها در مرجع شماره [46] معرفی گردیده‌اند:

۱. فقط ترمینال ۰ وجود خواهد داشت.
۲. وزن جمع‌شونده یال-۰ صفر می‌باشد.
۳. بزرگترین مقسوم‌علیه مشترک وزن‌های دیگر یال‌ها، یک می‌باشد.

٦-٦ نمایش بولی تغییریافته

در دیاگرام تصمیم‌گیری دودویی با یال‌های خصیصه‌دار، تنها توابع غیرمکمل به صورت مستقیم ساخته می‌شوند و سایر توابع از روی توابع غیر مکمل ساخته می‌شوند. برای این منظور مرجع [55] خصیصه‌ای به یال‌ها اضافه کرده است. در این نمایش، مکمل یک تابع بوسیله یک یال با خصیصه فعال که به آن تابع اشاره می‌کند، ساخته می‌شود.
نمایش دیاگرام بسط تیلور مثبت هم یک چنین ویژگی‌ای دارد. برای مکمل کردن تابع $f(x)$ ، $1-f(x)$ باید ساخته شود. از آنجا که وزن‌های جمع‌شونده در دیاگرام وجود دارد می‌توانیم عامل جمع‌شونده ۱ را فاکتور بگیریم، بنابراین $f(x)$ – باقی می‌ماند. به علاوه، از آنجا که وزن ضرب‌شونده داریم، ۱- را به عنوان عامل ضرب‌شونده فاکتور می‌گیریم و آنچه باقی می‌ماند $f(x)$ می‌باشد.

٦-٧ رابطه با دیگر دیاگرام‌ها

در این بخش شباهت‌ها و تفاوت‌های بین دیاگرام بسط تیلور مثبت[1] و دیاگرام تصمیم‌گیری دودویی[2]، دیاگرام تصمیم‌گیری دودویی با یال‌های وزن‌دار[1] و دیاگرام گشتاور دودویی ضربی کرانکر[2] بررسی خواهد شد. دیاگرام

[1] Taylor Expansion Diagram Plus
[2] Binary Decision Diagram

تصمیم‌گیری دودویی یک ساختمان‌داده برای نمایش عبارت‌های بولی می‌باشد. این دیاگرام از تجزیه شانون[3] استفاده می‌کند و هر گره دقیقا دو یال خروجی دارد. یال‌های دیاگرام تصمیم‌گیری دودویی وزن ندارند (نه جمع‌شونده و نه ضرب‌شونده). اثبات شده است که دیاگرام تصمیم‌گیری دودویی در نمایش توابع بولی بهینه می‌باشد.

دیاگرام تصمیم‌گیری دودویی با یال‌های وزن‌دار و دیاگرام گشتاور دودویی ضربی کرانکر ساختمان‌داده‌هایی برای نمایش توابع بولی و جبری درجه اول می‌باشند. دیاگرام تصمیم‌گیری دودویی با یال‌های وزن‌دار از رابطه ٦-٧ به عنوان تابع تجزیه استفاده می‌کند و یال‌هایش تنها وزن جمع‌شونده دارند. از طرف دیگر، دیاگرام گشتاور دودویی ضربی کرانکر از رابطه ٦-٨ و رابطه ٦-٩ به همراه رابطه ٦-٧ به عنوان تابع تجزیه استفاده می‌کند.

$$f(x) = f(0) + x(f(1) - f(0)) \qquad \textbf{رابطه ٦-٨}$$

$$f(x) = (1 - x)(f(0) - f(1)) + f(1) \qquad \textbf{رابطه ٦-٩}$$

در هر سطح، یکی از این توابع تجزیه استفاده می‌شود و انتخاب مناسب توابع تجزیه اثر زیادی در اندازه دیاگرام حاصل خواهد داشت. در دیاگرام گشتاور دودویی ضربی کرانکر، هر یال دارای وزن جمع‌شونده و وزن ضرب‌شونده می‌باشد.

٦-٧-١ دیاگرام تصمیم‌گیری دودویی و دیاگرام بسط تیلور مثبت

آشکار است که نمایش عبارات بولی در دیاگرام بسط تیلور مثبت و دیاگرام تصمیم‌گیری دودویی با یال‌های خصیصه‌دار[4] مشابه می‌باشند. اما دیاگرام بسط تیلور مثبت به دلیل داشتن وزن جمع‌شونده و ضرب‌شونده سربار[5] بیشتری دارد. از طرف دیگر، دیاگرام تصمیم‌گیری دودویی نمی‌تواند عبارات جبری را به طور مستقیم نمایش دهد. این عبارات‌ها باید به معادل بیتی خود شکسته شوند.

[1] Edge-Valued Binary Decision Diagram
[2] Kronecker Multiplicative Binary Moment Diagram
[3] Shannon
[4] Attributed Edge
[5] Overhead

٦-٧-٢ دیاگرام تصمیم‌گیری دودویی با یال‌های وزن‌دار و دیاگرام بسط تیلور مثبت

دیاگرام تصمیم‌گیری دودویی با یال‌های وزن‌دار از رابطه شانون (رابطه ٦-٧) برای نمایش توابع بولی و جبری درجه اول استفاده می‌کند. دیاگرام بسط تیلور مثبت نیز از تجزیه شانون برای این عبارت‌ها استفاده می‌کند (همه متغیرها در عبارت‌های بولی و جبری درجه اول دارای درجه یک می‌باشند). بنابراین در نمایش این عبارت‌ها، دیاگرام بسط تیلور مثبت و دیاگرام تصمیم‌گیری دودویی با یال‌های وزن‌دار از یک تابع تجزیه استفاده می‌کنند.
این دو دیاگرام دارای وزن جمع‌شونده می‌باشند. دیاگرام بسط تیلور مثبت دارای وزن ضرب‌شونده هم می‌باشد. بنابراین انتظار داریم که دیاگرام بسط تیلور مثبت فشرده‌تر از دیاگرام تصمیم‌گیری دودویی با یال‌های وزن‌دار باشد، البته با سربار بیشتر در یال‌ها (وزن‌های ضرب‌شونده).

٦-٧-٣ دیاگرام گشتاور دودویی ضربی کرانکر و دیاگرام بسط تیلور مثبت

دیاگرام گشتاور دودویی ضربی کرانکر از رابطه شانون به همراه رابطه ٦-٨ و رابطه ٦-٩ به عنوان تابع تجزیه استفاده می‌کند. دیاگرام بسط تیلور مثبت هم از دو تابع تجزیه استفاده می‌کند، تیلور و شانون. اختلاف دو دیاگرام در این است که انتخاب تابع تجزیه در دیاگرام گشتاور دودویی ضربی کرانکر دلخواه است (می‌توان تابع تجزیه را برای رسیدن به فشردگی بیشتر تغییر داد)، در حالیکه در دیاگرام بسط تیلور مثبت تابع تجزیه هر گره از درجه متغیر وابسته به آن گره به‌دست می‌آید. هر دو دیاگرام دارای وزن‌های جمع‌شونده و ضرب‌شونده می‌باشند. اگر توابع تجزیه دیاگرام گشتاور دودویی ضربی کرانکر را به تجزیه شانون محدود کنیم، عبارات بولی و جبری درجه اول به صورت مشابه توسط این دو دیاگرام نمایش پیدا می‌کنند.

٦-٨ نتایج تجربی

در این بخش نتایج تجربی که بر روی پردازنده پنتیوم ٤ با یک گیگابایت[1] حافظه بدست آمده است، ارائه می‌گردد. همه زمان‌ها برحسب ثانیه گزارش شده است. دیاگرام بسط تیلور و دیاگرام بسط تیلور مثبت با VisualC++ نسخه ٦ پیاده‌سازی گردیده‌اند و برای دیاگرام تصمیم‌گیری دودویی از بسته Cudd [56] که بسیار بهینه می‌باشد، استفاده شده است.

[1] Gigabyte

جدول ٦-١: مقایسه چندین ساختمان‌داده توسط مدارهای سطح گیت

Circuits	BDD		TED		TED+	
	Nodes	Time	Nodes	Time	Nodes	Time
Hamming 8bit	1445	0	1970	0.3	1445	0
Address Decoder 20bit	108	0	165	0	108	0
Parity Generator 9bit	329	0	384	0.1	329	0
Comparator 4bit	431	0	587	0.1	431	0
Comparator 7bit	940	0	963	0	940	0
Comparator 16bit	198484	0.3	198531	8.3	198484	4.1
Full Adder	30	0	37	0	30	0
Adding 9 CS 2bit each	4388	0	9663	84.3	4388	0.1
Adding 9 CS 4bit each	15194	0	Unfeasible	Unfeasible	15194	0.3
Adding 9 CS 8bit each	36750	0	Unfeasible	Unfeasible	36750	0.7
CLA Adder 8bit	9224	0	12398	13.8	9224	0.2
CLA Adder 16bit	2373630	2.7	Unfeasible	Unfeasible	2373630	52.3
Array Multiplier 8bit	184685	0.1	Unfeasible	Unfeasible	184685	2.5
Multiplier 4bit	1757	0	2008	0.3	1757	0
Multiplier 8bit	173091	0.1	Unfeasible	Unfeasible	173091	2.8
c17	34	0	38	0	34	0
c432	27005	0	Unfeasible	Unfeasible	27005	0.3
c499	510732	0.1	Unfeasible	Unfeasible	510732	2
c880	2501340	1.9	Unfeasible	Unfeasible	2501340	35.9
c1355	1733831	0.2	Unfeasible	Unfeasible	1733831	5
c1908	512736	0.3	Unfeasible	Unfeasible	512736	6.9

در سری اول آزمایش‌ها، جمع تعداد گره‌های مورد نیاز برای ساختن دیاگرام تصمیم‌گیری دودویی، دیاگرام بسط تیلور و دیاگرام بسط تیلور مثبت تعدادی محک[1] سطح گیت آورده شده است. جدول ٦-١ خلاصه‌ای از نتایج بدست آمده برای این محک‌ها را نشان می‌دهد. دیاگرام تصمیم‌گیری دودویی این مدارها حداقل ۳۰ گره دارد. ستونی که با BDD برچسب خورده است نتایج تبدیل این مدارها به دیاگرام تصمیم‌گیری دودویی را نشان می‌دهد، زیر ستون‌ها تعداد گره‌ها و زمان تبدیل را نشان می‌دهند. ستون‌های TED و +TED اطلاعات مربوط به دیاگرام بسط تیلور و دیاگرام بسط تیلور مثبت را نشان می‌دهند. همه دیاگرام‌ها برحسب ترتیب واحدی میان متغیرها، ساخته شده‌اند. لغت Unfeasible به معنای آن است که نمایش محک در ۱۲۰ ثانیه ساخته نشده است.

همیشه تعداد گره‌ها در دیاگرام بسط تیلور مثبت برابر با تعداد گره‌ها در دیاگرام تصمیم‌گیری دودویی می‌باشد. دیاگرام تصمیم‌گیری دودویی از نظر زمان تبدیل[2] بهتر از دیاگرام بسط تیلور مثبت می‌باشد. این امر به دلیل زمان

[1] Benchmark
[2] Conversion Time

مورد نیاز برای پردازش وزن‌های جمع‌شونده و ضرب‌شونده می‌باشد. زمان تبدیل دیاگرام بسط تیلور مثبت کمتر از دیاگرام بسط تیلور می‌باشد؛ چرا که در نمایش عبارات بولی، اندازه دیاگرام بسط تیلور بزرگتر است. به علاوه، پیچیدگی عملیات بولی در دیاگرام بسط تیلور مثبت کمتر از دیاگرام بسط تیلور می‌باشد (دیاگرام بسط تیلور مثبت در عملیات بولی مشابه دیاگرام تصمیم‌گیری دودویی می‌باشد).

جدول ٦-۲: مقایسه دیاگرام بسط تیلور معمولی و مثبت بوسیله معادلات جبری

Equations	TED		TED+	
	Nodes	Time	Nodes	Time
$\sum_{i=1}^{100} x_i$	102	0.1	101	2.6
$\prod_{i=1}^{100} x_i$	102	0.1	101	0.1
$\prod_{i=1}^{10}\sum_{j=1}^{10} x_{i,j}^{\ 2}$	102	0	101	0
$\prod_{i=1}^{20}\sum_{j=1}^{20} x_{i,j}^{\ 2}$	402	0.2	401	0.2
$\prod_{i=1}^{10}\sum_{j=1}^{10} x_{i,j}^{\ 3}$	102	0	101	0
$\prod_{i=1}^{20}\sum_{j=1}^{20} x_{i,j}^{\ 3}$	402	0.3	401	0.4
$\prod_{i=1}^{10}\sum_{j=1}^{10} x_{i,j}(i+j)$	102	0.1	101	0.1
$\prod_{i=1}^{20}\sum_{j=1}^{20} x_{i,j}(i+j)$	402	1.3	401	2.0
$\prod_{i=1}^{6}(\sum_{j=1}^{6} x_{i,j}^{\ j})^i$	38	0.6	37	0.9
$\prod_{i=1}^{8}(\sum_{j=1}^{8} x_{i,j}^{\ j})^i$	2382	19.5	2381	27.3
$(\sum_{i=1}^{10} x_i)x + \sum_{j=1}^{10} x_j$	23	0	76	0
$(\sum_{i=1}^{20} x_i)x + \sum_{j=1}^{20} x_j$	43	0	251	0.1

در همه این مدارها، واحدهای محاسباتی به لیستی از گیت‌ها سنتز[1] شده و سپس به دیاگرام‌های مربوط تبدیل شده‌اند.

مدارهایی که برای این آزمایش مورد استفاده قرار گرفته‌اند واحدهای محاسباتی واقعی و تعدادی از محک‌های ISCAS [57] می‌باشند. همه واحدهای محاسباتی، پایه‌ای برای واحدهای محاسباتی بزرگتر می‌باشند و بنابراین می‌توان نتایج را به همه واحدهای محاسباتی گسترش داد.

در سری دوم آزمایش‌ها، نمایش جبری دیاگرام بسط تیلور مثبت با دیاگرام بسط تیلور مقایسه شده است. عباراتی که برای این آزمایش مورد استفاده قرار گرفته‌اند، اغلب در طرح‌های انتقال‌ثبات مورد استفاده قرار می‌گیرند. نتایج این آزمایش در جدول ٦-٢ نشان داده شده است. در بیشتر موارد، تعداد گره‌های دیاگرام بسط تیلور مثبت یکی کمتر از دیاگرام بسط تیلور می‌باشد. این امر به دلیل تفاوت در نمایش ترمینال ۱ در این دو دیاگرام می‌باشد. به واقع، تعداد گره‌های متغیر در دو دیاگرام یکسان می‌باشد اما دیاگرام بسط تیلور دو گره ترمینال دارد در حالیکه دیاگرام بسط تیلور مثبت تنها یک ترمینال دارد (ترمینال ۱ با ترمینال ۰ و وزن جمع‌شونده یک ساخته می‌شود). بجز چند مورد که دیاگرام بسط تیلور مثبت گره‌های بیشتری دارد، تعداد گره‌ها در دو دیاگرام یکسان می‌باشد. زمان تبدیل در دیاگرام بسط تیلور مثبت بیشتر از دیاگرام بسط تیلور می‌باشد؛ چرا که پیچیدگی الگوریتم‌ها در دیاگرام بسط تیلور مثبت بیشتر از دیاگرام بسط تیلور می‌باشد.

در سری سوم آزمایش‌ها، یک پردازنده به دیاگرام بسط تیلور و دیاگرام بسط تیلور مثبت تبدیل گردیده است. این پردازنده گذرگاه داده و آدرس ١٦ بیتی دارد. هفت دستورالعمل این پردازنده در جدول ٦-٣ نشان داده شده است.

جدول ٦-٣: مجموعه دستورالعمل‌های پردازنده

Opcode	Instruction
000	LDA
001	STA
010	ADD
011	SUB
100	JMP
101	JEZ
110	HLT

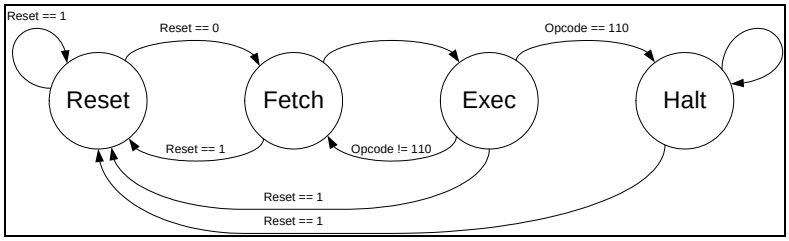
[1] Synthesis

واحد محاسبات جبری و منطقی[1] این پردازنده دارای دو ورودی A و B و یک ورودی دو بیتی است که یکی از چهار عملیات مختلف را انتخاب می‌کند. جدول ٦-٤ عملیات مختلف این واحد محاسباتی را نشان می‌دهد.

جدول ٦-٤: عملیات واحد محاسبه و منطق

ALU_op	ALUout
00	B
01	A - B
10	A + B
11	B + 1

واحد کنترل که در شکل ٦-٤ نشان داده شده است ماشینی با چهار حالت می‌باشد. بیشتر دستورالعمل‌ها به دو پالس ساعت برای اجرا نیاز دارند. دستورالعمل Halt پردازنده را متوقف می‌کند تا زمانی که Reset فعال شود.



شکل ٦-٤: واحد کنترل پردازنده

واحد کنترل به رابطه انتقال[2] بین حالت‌ها و توابع خروجی تبدیل گشته و سپس این توابع به دیاگرام بسط تیلور و دیاگرام بسط تیلور مثبت تبدیل شده‌اند. به صورت همزمان مسیرداده[3] این پردازنده هم به این دیاگرام‌ها تبدیل شده است. نتایج تبدیل این پردازنده در جدول ٦-٥ نشان داده شده است. این جدول نشان می‌دهد که در مقایسه با دیاگرام بسط تیلور، دیاگرام بسط تیلور مثبت ۳٦٪ از نظر تعداد گره و دو برابر از نظر زمان تبدیل، بهتر عمل کرده است.

جدول ٦-٥: مقایسه دیاگرام بسط تیلور معمولی و مثبت بوسیله یک پردازنده ساده

	Nodes	Time
TED	175	0.031
TED+	112	0.015

[1] Arithmetic Logic Unit
[2] Transition Relation
[3] Datapath

اگرچه این آزمایش روی یک پردازنده ساده انجام شده است، اما این پردازنده مسیرداده و واحد کنترل دارد که معمول همه طرح‌های بزرگ می‌باشد. به علاوه, این مثال شامل بخش‌های بولی و جبری می‌باشد. بنابراین این مثال نشان‌دهنده کارایی دیاگرام بسط تیلور مثبت در نمایش پردازنده‌های پیچیده می‌باشد.

٦-٩ نتیجه‌گیری

در این فصل، دیاگرام بسط تیلور مثبت ارائه گردید. در دیاگرام بسط تیلور مثبت تابع تجزیه گره‌های با دو یال خروجی به شانون[1] تغییر داده شد. این امر به کاهش کارایی نمایش عبارات جبری منجر گردید. برای افزایش کارایی، وزن‌های جمع‌شونده (علاوه بر وزن‌های ضرب‌شونده) به یال‌ها اضافه شدند. به این طریق یک ساختمان‌داده جدید برای نمایش طرح‌های جبری و بولی مختلط[2] ایجاد گردید. دیاگرام بسط تیلور مثبت در نمایش عبارات بولی مشابه دیاگرام تصمیم‌گیری دودویی با یال‌های خصیصه‌دار می‌باشد. از طرف دیگر، دیاگرام بسط تیلور مثبت در نمایش بعضی عبارات جبری اندکی ضعیف‌تر از دیاگرام بسط تیلور می‌باشد. به دلیل آنکه احتمال وجود چنین عباراتی در سیستم‌های دیجیتال کم می‌باشد و از آنجا که بخش‌های بولی در طرح‌های واقعی بزرگتر از بخش‌های جبری هستند، کارایی دیاگرام بسط تیلور مثبت بیشتر از دیاگرام بسط تیلور می‌باشد. نتایج تجربی نشان می‌دهد که بسیاری از عبارات جبری کاربردی که به صورت معمول در توصیفات انتقال‌ثبات به کار برده می‌شوند، به خوبی بوسیله دیاگرام بسط تیلور مثبت نمایش پیدا می‌کنند. بنابراین برای نمایش طرح‌هایی که شامل بخش‌های سطح گیت و سطح انتقال‌ثبات (طول بیت ۱ و بیشتر) می‌باشند، دیاگرام بسط تیلور مثبت یک راه‌حل مناسب می‌باشد.

[1] Shannon
[2] Mixed

فصل هفتم: دیاگرام بسط تیلور تقویت‌شده

در این فصل یک ساختمان‌داده یگانی[1] برای نمایش طرح‌های انتقال‌ثبات ارائه می‌گردد. در حال حاضر نمایش یگانی و مبتنی بر گرافی به نام دیاگرام بسط تیلور[2] وجود دارد. اگرچه می‌توان دیاگرام بسط تیلور را برای نمایش بهینه عبارات جبری در سطح‌کلمه به کار گرفت، اما آنها در نمایش عبارات بولی سطح بیت بهینه نمی‌باشند. همچنین دیاگرام بسط تیلور نمی‌تواند عبارات بولی در سطح‌کلمه (بردار) را نمایش دهد. در این فصل تغییراتی به دیاگرام بسط تیلور اعمال می‌گردد که قابلیت نمایش سطح بیت آنرا افزایش می‌دهد و همزمان قابلیت نمایش عبارات بولی در سطح‌کلمه را به آن اضافه می‌کند.

۷-۱ مقدمه

با افزایش پیچیدگی و اندازه سیستم‌های دیجیتال لازم است که ارزیابی طرح‌ها در مراحل آغازین چرخه طراحی[3] صورت گیرد. این امر نیازمند ابزارهای ارزیابی خودکار[4] در سطوح بالا (سطح انتقال‌ثبات[5] یا سطح رفتاری[6]) می‌باشد.

هم اکنون ابزارهایی برای ارزیابی‌رسمی طرح‌ها در سطوح انتزاع پایین مثل سطح گیت وجود دارد؛ اما ابزارهایی که بتوانند توصیفات سطح بالا را ارزیابی کنند، هنوز به سطح قابل قبولی نرسیده‌اند. این مساله تا قسمتی از نبود یک نمایش مناسب برای طرح‌های انتقال‌ثبات ناشی شده است. از آنجا که دیاگرام تصمیم‌گیری دودویی[7] [13] و مشتقاتش [21][25] طرح‌های در سطح گیت را به نحو موثری نمایش می‌دهند، بسیاری از ابزارهای ارزیابی، توصیف را سنتز می‌کنند تا بتوانند از مزایای دیاگرام تصمیم‌گیری دودویی که در سطح گیت می‌باشد، بهره ببرند. هدف این فصل ارائه یک نمایش مبتنی بر گراف برای توصیفات انتقال‌ثبات می‌باشد. برای رسیدن به این هدف از دیاگرام بسط تیلور [32][33][34] به عنوان مبنای نمایش استفاده شده است. دیاگرام بسط تیلور مزایای زیادی نسبت به سایر نمایش‌های مبتنی بر گراف دارا می‌باشد، اما دارای چندین محدودیت نیز هست. دیاگرام بسط تیلور را تقویت خواهیم کرد تا بر محدودیت‌هایش غلبه کند. ساختمان‌داده حاصل را دیاگرام بسط تیلور تقویت‌شده[8] می‌نامیم. نتایج تجربی بدست آمده نشان می‌دهد که دیاگرام بسط تیلور تقویت‌شده برای نمایش طرح‌های انتقال‌ثبات مناسب می‌باشد.

---

[1] Canonical
[2] Taylor Expansion Diagram
[3] Design Cycle
[4] Automatic
[5] Register Transfer
[6] Behavioral
[7] Binary Decision Diagram
[8] Enhanced Taylor Expansion Diagram

۷-۲ کارهای قبلی

توابع بولی را اغلب بوسیله دیاگرام‌های تصمیم‌گیری نمایش می‌دهند. دیاگرام تصمیم‌گیری دودویی مرتب‌شده [13] پرکاربردترین دیاگرام تصمیم‌گیری در کاربردهای خودکارسازی طراحی الکترونیک[1] می‌باشد [15]. دیاگرام تصمیم‌گیری دودویی مرتب‌شده و مشتقاتش [21][25] به صورت موفقیت‌آمیزی در نمایش طرح‌های سطح گیت به کار گرفته شدند؛ اما محدودیت‌هایی در نمایش مدارهای محاسباتی دارند.

برای نمایش مدارهای محاسباتی، دیاگرام‌های تصمیم‌گیری سطح‌کلمه به کار گرفته می‌شوند. نمونه‌هایی از دیاگرام‌های تصمیم‌گیری سطح‌کلمه در زیر آورده شده است: دیاگرام تصمیم‌گیری دودویی با ترمینال‌های متعدد[2] [17][10]، دیاگرام تصمیم‌گیری دودویی با یال‌های وزن‌دار[3] [58]، دیاگرام گشتاور دودویی ضربی[4] [16]، دیاگرام تصمیم‌گیری ترکیبی[5] [45] و دیاگرام گشتاور دودویی ضربی کرانکر[6] [46][47][48]. این دیاگرام‌ها، نمایش مبتنی بر گراف توابعی با دامنه بولی و برد عددی می‌باشند. بنابراین توابع محاسباتی باید به معادل بیتی خود تبدیل شوند تا بوسیله این دیاگرام‌های تصمیم‌گیری سطح‌کلمه نمایش پیدا کنند.

با این وجود، با افزایش پیچیدگی سیستم‌های دیجیتال، نیاز به سطوح انتزاع بالاتر بیشتر احساس شد. دیاگرام بسط تیلور [32][33][34] به عنوان پاسخی بر این نیاز ارائه گردید. دیاگرام بسط تیلور می‌تواند توابعی با دامنه و برد عددی را نمایش دهد. بنابراین برخلاف دیاگرام‌های تصمیم‌گیری سطح‌کلمه، یک تابع محاسباتی نباید به معادل بیتی خود تبدیل شود تا نمایش پیدا کند.

اگرچه دیاگرام بسط تیلور مزایای بسیاری نسبت به دیاگرام‌های تصمیم‌گیری سطح‌کلمه دارد، اما دارای محدودیت‌هایی هم هست. در این فصل این محدودیت‌ها را برطرف می‌کنیم تا به یک نمایش مناسب برای سطح انتقال‌ثبات دست پیدا کنیم.

۷-۳ دیاگرام بسط تیلور

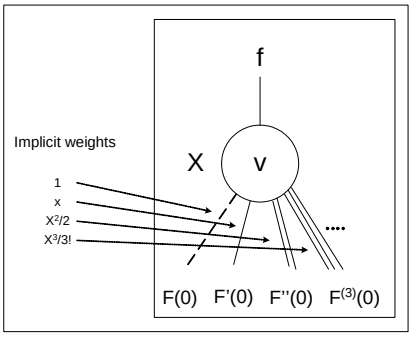
دیاگرام بسط تیلور یک نمایش مبتنی بر گراف است که از سری تیلور به عنوان قاعده تجزیه[7] استفاده می‌کند [32][33][34]. سری تیلور تابع مشتق‌پذیر $f(x)$ حول $x=0$ به صورت زیر می‌باشد:

[1] Electronic Design Automation
[2] Multi Terminal Binary Decision Diagram
[3] Edge-Valued Binary Decision Diagram
[4] Multiplicative Binary Moment Diagram
[5] Hybrid Decision Diagram
[6] Kronecker Multiplicative Binary Moment Diagram
[7] Decomposition Method

$$f(x) = f(0) + xf'(0) + \frac{x^2}{2!} f''(0) + \frac{x^3}{3!} f'''(0) + \cdots$$ **رابطه ۷-۱**

که $f'(0)$، $f''(0)$ و $f'''(0)$ مشتقات اول، دوم و سوم تابع f حول $x = 0$ می‌باشند. تجزیه تابع برحسب رابطه ۷-۱ به صورت بازگشتی ادامه پیدا می‌کند.

دیاگرام بسط تیلور سه گونه گره دارد: گره سطح بیت، گره سطح بردار و گره ترمینال. گره‌های سطح بیت و بردار نشان دهنده متغیرها بوده و گره‌های ترمینال نشان دهنده ثابت‌ها می‌باشند. اختلاف بین بیت و بردار آن است که $A \times A = A^2$ اگر A یک بردار باشد در حالیکه $a \times a = a$ اگر a یک بیت باشد. هرگره در دیاگرام بسط تیلور یک برچسب دارد که متغیر وابسته[1] به آن گره را مشخص می‌کند. مثل همه دیاگرام‌های تصمیم‌گیری یگانی (مثلا دیاگرام تصمیم‌گیری دودویی مرتب‌شده)، متغیرهای دیاگرام بسط تیلور، مرتب‌شده هستند. تابع یک گره بوسیله سری تیلور مطابق رابطه ۷-۱ بدست می‌آید. درجه خروجی یک گره به درجه خروجی متغیر وابسته به آن گره بستگی دارد. درجه خروجی گره‌های ترمینال صفر است.



شکل ۷-۱: تجزیه در دیاگرام بسط تیلور

شکل ۷-۱ تجزیه تابع f برحسب متغیر x را نشان می‌دهد. در این فصل به مشتق k–ام یک تابع در یک گره، فرزند k–ام آن گره گفته می‌شود: $f(x=0)$ فرزند-۰، $f'(x=0)$ فرزند-۱، $f''(x=0)$ فرزند-۲ و .... همچنین به یال‌های متناظر یال-۰ (خط چین)، یال-۱ (خط ممتد)، یال-۲ (خط دوگانه) و ... گفته می‌شود. از بسط تیلور آشکار است که هر یال یک عامل ضربی پنهان[2] دارد: x^0 برای یال-۰، x^1 برای یال-۱، $\frac{x^2}{2!}$ برای

[1] Associated Variable
[2] Implicit Multiplicative

یال-۲ و .... به علاوه، هر یال در دیاگرام بسط تیلور یک وزن ضرب‌شونده دارد که از بسط تیلور محاسبه می‌گردد.

دیاگرام بسط تیلور می‌تواند توابع بولی در سطح بیت را هم نمایش دهد. برای نمایش عبارات بولی از روابط زیر استفاده می‌شود (x و y متغیرهای بیتی می‌باشند) [32][33][34].

رابطه ۷-۲
$$NOT(x) = 1 - x$$

رابطه ۷-۳
$$AND(x, y) = x \times y$$

رابطه ۷-٤
$$OR(x, y) = x + y - x \times y$$

۷-٤ نمایش مناسب در سطح انتقال‌ثبات

برای نمایش طرح‌های انتقال‌ثبات به یک نمایش مناسب نیاز می‌باشد. در این بخش به ویژگی‌های ساختارهای انتقال‌ثبات می‌پردازیم.

۷-٤-۱ نیازهای سطح انتقال‌ثبات

در این بخش نیازهای سطح انتقال‌ثبات از سه جهت مختلف بررسی می‌شوند. این نیازها عبارتند از پذیرفتن متغیرهای سطح بیت و بردار، پذیرفتن متغیرهای بولی و جبری و نمایش بهینه عبارات بولی بیتی. این نیازها در این بخش بررسی می‌گردند.

۷-٤-۱-۱ متغیرهای انتقال‌ثبات

در بیشتر طرح‌های سطح انتقال‌ثبات بردارها و بیت‌ها همزمان وجود دارند. یک جمع کننده هشت بیتی با دو ورودی برداری A و B و یک ورودی بیتی c_i را در نظر بگیرید. توصیف سطح انتقال‌ثبات این جمع‌کننده عبارت است از $A+B+c_i$. بسیاری از دیاگرام‌های تصمیم‌گیری نمی‌توانند بردارها را نمایش دهند و از این رو بردارها را به اجزا بیتی‌شان می‌شکنند. در این مثال، یک دیاگرام تصمیم‌گیری معمولی برای نمایش عمل جمع، متغیرهای برداری A و B را به هشت متغیر یک بیتی می‌شکند. بنابراین این نمایش‌ها با توصیفات سطح انتقال‌ثبات سازگار نبوده و متمایل به نمایش‌های سطح بیت می‌باشند. لازم است که یک نمایش انتقال‌ثبات، بردارها را به صورت یک کل و نه به عنوان مجموعه‌ای از بیت‌ها نمایش دهد.

در بسیاری از موارد نمایش بیت و بردار متفاوت می‌باشد. برای مثال اگر A یک بردار باشد، $A \times A = A^2$ در حالیکه اگر a یک بیت باشد، $a \times a = a$. بنابراین نتیجه می‌گیریم که ساختمان‌داده‌های سطح بیت فعلی را

نمی‌توان برای نمایش بهینه متغیرها در سطح انتقال‌ثبات به کار گرفت. همچنین لازم است که بردارها و بیت‌های مختلط[1] را همزمان نمایش داد (همانطور که مثال جمع‌کننده پیشنهاد می‌کند).

۷-٤-۱-۲ عملگرهای سطح انتقال‌ثبات

عملگرهای مختلفی در طرح‌های سطح انتقال‌ثبات مورد استفاده قرار می‌گیرند. عملگرهای جبری و بولی از پرکاربردترین عملگرهای سطح انتقال‌ثبات می‌باشند. این عملگرها در جدول ۷-۱ آورده شده‌اند. نمایش این عملگرها برای ساختمان‌داده مورد استفاده در ارزیابی‌رسمی سطح انتقال‌ثبات ضروری می‌باشد.

جدول ۷-۱: عملگرهای سطح انتقال‌ثبات

Arithmetic Operators	+ − ×
Boolean Operators	& | ! ∧

این عملگرها به بیت‌ها و بردارها اعمال می‌گردند و ویژگی‌های متمایزی دارند. عملگرهای جبری، هر بردار را یک موجودیت واحد در نظر می‌گیرند، در حالیکه عملگرهای بولی به صورت بیت به بیت به بردارها اعمال می‌شوند. این خصوصیات باید بوسیله یک ساختمان‌داده یکتا پشتیبانی شود.

برای ساختمان‌داده سطح بیت، توانایی نمایش عملگرهای جبری یا بولی کافی می‌باشد؛ زیرا عملگرهای بولی را می‌توان به کمک عملگرهای جبری ساخت و برعکس. برای مثال، دیاگرام تصمیم‌گیری دودویی عملگرهای بولی و دیاگرام گشتاور دودویی عملگرهای جبری در سطح بیت را پشتیبانی می‌کنند و هر دو همه عملگرهای سطح بیت را نمایش می‌دهند. از طرف دیگر برای نمایش سطح‌کلمه (بردار) ساختمان‌داده‌ای که فقط عملگرهای جبری یا بولی را پشتیبانی کند، کفایت نمی‌کند. این امر به دلیل آن است که عملگرهای بولی در سطح‌کلمه (بردار) را نمی‌توان از روی عملگرهای جبری ساخت مگر آنکه بردارها به بیت‌ها تجزیه شوند و برعکس. یک مثال برای این بحث عملگر ضرب می‌باشد که در سطح بیت مشابه عملگر & عمل می‌کند؛ اما در سطح‌کلمه نه تنها یکی نمی‌تواند جای دیگری را بگیرد بلکه هیچ ارتباطی هم میان آن دو وجود ندارد.

یک ساختمان‌داده مرتبط با توصیفات انتقال‌ثبات باید متغیرها و عملگرهای سطح بیت و بردار را پشتیبانی کند. اما عملگرهای بولی در سطح‌کلمه (بردار) به نمایش بولی و عملگرهای جبری به نمایش جبری نیاز دارند.

[1] Mixed

۷-٤-۱-۳ نمایش بهینه در سطح بیت

علاوه بر مباحث بالا، بسیاری از طرح‌های سطح انتقال‌ثبات بخش‌های بولی سطح بیت بزرگی دارند. برای نمونه، بسیاری از ریزپردازنده‌ها یک مسیرداده[1] و چندین واحد کنترل دارند. واحد کنترل سیگنال‌های سطح بیتی دارد تا عملگرهای مسیرداده را کنترل نماید. واحد کنترل معمولا بوسیله عبارات بولی توصیف می‌شود. بنابراین نمایش بهینه عملیات بولی در سطح بیت از نیازهای اساسی ساختمان‌داده مورد استفاده در طرح‌های انتقال‌ثبات می‌باشد.

۷-٤-۲ ساختمان‌داده‌های موجود

در این بخش چندین ساختمان‌داده که در نمایش طرح‌های انتقال‌ثبات مورد استفاده قرار می‌گیرند، بررسی می‌گردند.

۷-٤-۲-۱ دیاگرام تصمیم‌گیری دودویی

هرچند که دیاگرام تصمیم‌گیری دودویی نمایش سطح بیت خوبی دارد؛ اما نمی‌تواند بردارها را نمایش دهد. همه بردارها باید به بیت‌ها شکسته شوند تا نمایش داده شوند. به علاوه همه عملگرها، بوسیله عملگرهای بولی سطح بیت نمایش داده شوند. برای مثال، عملگر ضرب باید بوسیله عملگرهای بولی نشان داده شود. به دلیل عدم پشتیبانی از بردارها و عملگرهای جبری، دیاگرام تصمیم‌گیری دودویی راه‌حل مناسبی برای نمایش سطح انتقال‌ثبات نمی‌باشد.

۷-٤-۲-۲ دیاگرام‌های تصمیم‌گیری سطح‌کلمه

دیاگرام‌های تصمیم‌گیری سطح‌کلمه چندین ساختمان‌داده مختلف از قبیل دیاگرام تصمیم‌گیری دودویی با ترمینال‌های متعدد، دیاگرام تصمیم‌گیری دودویی با یالهای وزن‌دار، دیاگرام گشتاور دودویی ضربی و دیاگرام گشتاور دودویی ضربی کرانکر را در بر می‌گیرند. مشابه دیاگرام تصمیم‌گیری دودویی، دیاگرام‌های تصمیم‌گیری سطح‌کلمه نمی‌توانند بردارها را نمایش دهند. تمایز دیاگرام‌های تصمیم‌گیری سطح‌کلمه و دیاگرام تصمیم‌گیری دودویی در پشتیبانی از عملگرهای جبری در سطح بیت می‌باشد؛ در حالیکه دیاگرام تصمیم‌گیری دودویی عملگرهای جبری را به معادل بولی خود تبدیل می‌کند. بسیاری از دیاگرام‌های تصمیم‌گیری سطح‌کلمه، برای اشتراک بیشتر زیرگراف‌ها، وزن‌هایی در یال‌های خود دارند. این وزن‌ها، ثابت‌ها را از عبارات جبری جدا

[1] Datapath

می‌کنند و باعث می‌شوند که بخش‌های باقیمانده از این عبارات، قابلیت اشتراک بیشتری داشته باشند. نمایش عملگرهای جبری در سطح بیت و وزن یال‌ها این ساختمان‌داده‌ها را برای نمایش سطح انتقال‌ثبات فشرده‌تر از دیاگرام تصمیم‌گیری دودویی می‌سازند. اما بدلیل عدم پشتیبانی از بردارها، این ساختمان‌های داده برای نمایش انتقال‌ثبات مناسب نمی‌باشند.

۷-٤-۲-۳ دیاگرام بسط تیلور

برخلاف دیاگرام تصمیم‌گیری دودویی و دیاگرام‌های تصمیم‌گیری سطح‌کلمه، دیاگرام بسط تیلور می‌تواند بیت‌ها و بردارها را نمایش دهد. از آنجا که عملگرهای بولی در سطح بیت را می‌توان بوسیله عملگرهای جبری توصیف کرد (رابطه ۷-۲ تا رابطه ۷-٤)، دیاگرام بسط تیلور می‌تواند عملگرهای بولی را نمایش دهد. اما بدلیل آنکه بین عملگرهای جبری و بولی در سطح بردار ارتباطی وجود ندارد، دیاگرام بسط تیلور نمی‌تواند عبارات بولی در سطح بردار را نمایش دهد. تنها راه ممکن برای نمایش عملگرهای بولی برداری، شکستن آنها به معادل بیتی‌شان و اعمال رابطه ۷-۲ تا رابطه ۷-٤ به بیت‌های حاصل می‌باشد. آشکار است که عملگرهای بولی برداری به دفعات در مشخصه‌های انتقال‌ثبات به کار برده می‌شوند. به علاوه، هنگامی که عملگرهای جبری و بولی به صورت همزمان به یک بردار اعمال می‌گردند، بردار باید به بیت‌هایش تجزیه شود و قابلیت‌های دیاگرام بسط تیلور نمی‌تواند مورد استفاده قرار گیرد. برای مثال، فرض کنید که واحد محاسباتی ۸ بیتی یک پردازنده ساده دو عمل + و & انجام می‌دهد. ورودی انتخابگر این مدار (sel)، بین عملیات + و & یکی را انتخاب می‌کند. برای این مدار، عبارت زیر باید بوسیله دیاگرام بسط تیلور نمایش داده شود.

$$(1-sel)\times(A+B)+sel\times(A\&B)$$ **رابطه ۷-۵**

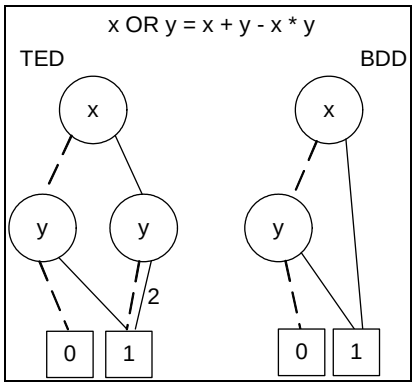
از آنجا که دیاگرام بسط تیلور نمی‌تواند $A\&B$ را در سطح بردار نمایش دهد، ورودی‌های A و B باید به بیت‌های خود شکسته شوند. شکسته شدن بیت‌های A و B باعث می‌شود که نمایش $A+B$ نیز در سطح بیت صورت گیرد و کارایی دیاگرام بسط تیلور تا حد دیاگرام گشتاور دودویی[1] کاهش یابد. بنابراین برای بسیاری از طرح‌های انتقال‌ثبات، دیاگرام بسط تیلور نمی‌تواند بهتر از دیاگرام گشتاور دودویی عمل نماید. جدول ۷-۲ قابلیت‌های دیاگرام تصمیم‌گیری دودویی، دیاگرام‌های تصمیم‌گیری سطح‌کلمه و دیاگرام بسط تیلور را در نمایش سطح بیت و بردار جمع‌بندی می‌کند.

[1] Binary Moment Diagram

جدول ۷-۲: مقایسه دیاگرام تصمیم‌گیری دودویی، دیاگرام‌های تصمیم‌گیری سطح‌کلمه و دیاگرام بسط تیلور

	Arithmetic		Boolean	
	Bit	Vector	Bit	Vector
BDDs	Y	N	Y	N
WLDDs	Y	N	Y	N
TEDs	Y	Y	Y	N

یکی دیگر از محدودیت‌های دیاگرام بسط تیلور نمایش ضعیف توابع بولی در سطح بیت می‌باشد. معمولا نمایش یک تابع بولی بوسیله دیاگرام تصمیم‌گیری دودویی بهینه‌تر از نمایش دیاگرام بسط تیلور می‌باشد. شکل ۷-۲ این ناکارایی دیاگرام بسط تیلور را نشان می‌دهد. همانطور که مشخص است، نمایش دیاگرام بسط تیلور تابع OR سه گره دارد در حالیکه نمایش دیاگرام تصمیم‌گیری دودویی این تابع فقط دو گره دارد. این ناکارایی می‌تواند برای عبارات بولی بزرگتر، بیشتر شود. همانطور که در بخش ۷-٤-۳-۱ عنوان شد، قابلیت نمایش بهینه بولی در سطح بیت برای ساختمان‌داده انتقال‌ثبات ضروری می‌باشد.



شکل ۷-۲: نمایش دیاگرام تصمیم‌گیری دودویی و دیاگرام بسط تیلور تابع $x OR y$

علی‌رغم آنکه دیاگرام بسط تیلور در نمایش سطح انتقال‌ثبات قوی‌تر از دیاگرام تصمیم‌گیری دودویی و دیاگرام‌های تصمیم‌گیری سطح‌کلمه عمل می‌کند، در نمایش سطح انتقال‌ثبات دارای نقاط ضعفی نیز می‌باشد. بنابراین ضروری است که برای رسیدن به ساختمان‌داده‌ای کارا در سطح انتقال‌ثبات دیاگرام بسط تیلور، بهبود بخشیده شود.

۷-۵ راه‌حل پیشنهادی

در این بخش دیاگرام بسط تیلور را تغییر می‌دهیم تا اشکالات عنوان شده پیشین برطرف گردند.

۷-۵-۱ نمایش بهینه عبارات بولی در سطح بیت

نمایش توابع بولی در سطح بیت یکی از اشکالات دیاگرام بسط تیلور می‌باشد. این به معنای آن است که نمایش دیاگرام بسط تیلور توابع بولی معمولا بزرگتر از نمایش دیاگرام تصمیم‌گیری دودویی همان توابع می‌باشد.
دیاگرام بسط تیلور تقویت‌شده که در پایین مورد بحث قرار می‌گیرد بر این حقیقت استوار است که نمایش دیاگرام بسط تیلور در بعضی مواقع هنگامی که متغیر تجزیه از درجه یک باشد، می‌تواند ساده شود. این امر می‌تواند نمایش توابع بولی سطح بیت را بسیار ساده سازد. همانطور که در بخش قبلی عنوان گردید، دیاگرام بسط تیلور در هر سطح یک تابع را به سری تیلور آن تابع (رابطه ۷-۱) تجزیه می‌کند. هنگامی که درجه متغیر تجزیه یک باشد سری تیلور برابر خواهد بود با

رابطه ۷-٦

$$f(x) = f(0) + xf'(0)$$

که رابطه زیر در آن صدق می‌کند.

رابطه ۷-۷

$$f'(0) = f(1) - f(0)$$

با استفاده از رابطه ۷-۷ در رابطه ۷-٦ و جابجایی متغیرها، رابطه ۷-۸ بدست می‌آید.

رابطه ۷-۸

$$f(x) = (1-x) f(0) + xf(1)$$

آشکار است که رابطه ۷-۸ شبیه تجزیه شانون[1] می‌باشد که به عنوان تابع تجزیه در دیاگرام تصمیم‌گیری دودویی استفاده می‌شود.

برای بهبود نمایش منطقی سطح بیت دیاگرام بسط تیلور، در گره‌های سطح بیت که همیشه دو یال خروجی دارند (همیشه درجه یک می‌باشند) از رابطه ۷-۸ بجای رابطه ۷-٦ به عنوان تابع تجزیه استفاده می‌کنیم. با انجام این کار و استفاده از رابطه ۷-۲، رابطه ۷-۳ و رابطه ۷-٤ به عنوان NOT، AND و OR، دیاگرام بسط تیلور تقویت‌شده در نمایش بولی سطح بیت مشابه دیاگرام تصمیم‌گیری دودویی عمل خواهد کرد. از آنجا که به صورت سراسری در همه گره‌های بیتی از رابطه ۷-۸ استفاده می‌شود، دیاگرام بسط تیلور تقویت‌شده یگانی باقی خواهد ماند.

[1] Shannon

نشان داده شده است که رابطه زیر بین نمایش بولی دیاگرام بسط تیلور و دیاگرام تصمیم‌گیری دودویی وجود دارد [59].

- توابع بولی‌ای وجود دارند که نمایش دیاگرام تصمیم‌گیری دودویی با اندازه چندجمله‌ای برای آنها وجود دارد اما اندازه همه نمایش‌های دیاگرام بسط تیلورشان نمایی می‌باشد.
- توابع بولی‌ای وجود دارند که نمایش دیاگرام بسط تیلور با اندازه چندجمله‌ای برای آنها وجود دارد اما اندازه همه نمایش‌های دیاگرام تصمیم‌گیری دودویی‌شان نمایی می‌باشد.

اما در عمل، نتایج تجربی نشان می‌دهد که دیاگرام تصمیم‌گیری دودویی در نمایش بخش‌های بولی سطح بیت بسیاری از طرح‌های انتقال‌ثبات فشرده‌تر از دیاگرام بسط تیلور می‌باشد. در بخش نتایج تجربی، این واقعیت نشان داده می‌شود.

۷-۵-۲ عملگرهای بولی سطح بردار

همانطور که در بخش‌های قبلی عنوان گردید، دیاگرام بسط تیلور نمی‌تواند عملگرهای بولی سطح بردار را نمایش دهد. در این بخش تغییراتی به دیاگرام بسط تیلور اعمال می‌گردد که قابلیت نمایش چنین عملگرهایی را به آن اضافه کند. به این نمایش، دیاگرام بسط تیلور تقویت‌شده گفته می‌شود.

برای نمایش عملگرهای بولی سطح بردار، مشابه عملگرهای بولی سطح بیت سعی خواهد شد تا آنها را بوسیله معادلات جبری توصیف نمود. برخلاف عملگرهای بولی سطح بیت، عملگرهای بولی سطح بردار را نمی‌توان کاملا با معادلات جبری توصیف کرد. اما عبارات حاصل، عملگرهای بولی ساده‌تری دارند. برای این منظور از روابط زیر استفاده می‌شود (A و B بردار می‌باشند).

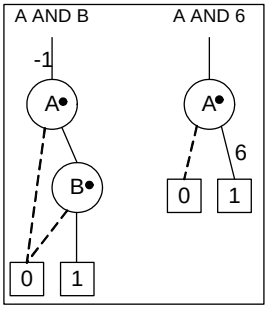
$$NOT(A) = -1 - A \qquad \textbf{رابطه ۷-۹}$$

$$AND(A, B) = A \,\&\, B \qquad \textbf{رابطه ۷-۱۰}$$

$$OR(A, B) = A + B - A \,\&\, B \qquad \textbf{رابطه ۷-۱۱}$$

به کمک روابط بالا، عملگرهای بولی سطح بردار به عملگرهای جبری و عملگر برداری & تبدیل می‌شوند. بخش‌های جبری این روابط بوسیله دیاگرام بسط تیلور قابل نمایش می‌باشند. در ادامه این بخش نشان خواهیم داد که چگونه عملگر برداری & را نمایش دهیم. پس از آن نشان داده خواهد شد که چگونه عباراتی با بخش‌های جبری و بولی را در یک ساختمان‌داده واحد نمایش داد. برای نمایش عملگر برداری & از تجزیه

شانون تعمیم‌یافته[1] استفاده می‌کنیم. تجزیه شانون تعمیم‌یافته می‌تواند عملگرهای برداری & را نمایش دهد و از قرار دادن مقدار همه-۰ برای متغیر وابسته به یک گره، فرزند-۰ بدست می‌آید و مقدار همه-۱ برای متغیر آن گره فرزند-۱ را بوجود می‌آورد. مثال‌هایی از نمایش عملگر برداری & در شکل ۷-۳ نشان داده شده است. در این شکل A و B متغیرهای ۸ بیتی هستند. در هر مورد، در گره‌های ترمینال مقدار عددی به معادل بیتی خود تبدیل می‌شود. برای تناظر با Verilog اعداد به صورت مکمل-۲ نگهداری می‌شوند. برای مثال مقدار همه-۱ (۱۱۱۱۱۱۱۱) برابر با ۱- می‌گردد که در یکی از یال‌های شکل ۷-۳ نشان داده شده است. دایره توپر داخل گره‌های این شکل، بیانگر آن است که این گره‌ها عبارات بولی برداری & را نمایش می‌دهند و نه عبارات جبری معمولی را.



شکل ۷-۳: نمایش بولی برداری & دیاگرام بسط تیلور تقویت‌شده

مباحث عنوان شده در بالا به ما اجازه می‌دهد که عملگرهای جبری و عملگر برداری & را نمایش دهیم. در ادامه به نمایش همزمان عبارات جبری و بولی می‌پردازیم. برای این منظور، معادلات بولی برداری با استفاده از رابطه ۷-۹ تا رابطه ۷-۱۱ به معادلات جبری و عملگر برداری & تبدیل می‌شوند. بعد از انجام این مرحله، معادلات مختلط حاصل را به حداقل تعداد عباراتی تبدیل می‌کنیم به نحوی که هر عبارت یا جبری بوده و یا فقط شامل عملگر & باشد.

پس از انجام این کار، بخش جبری به کمک دیاگرام بسط تیلور و عبارات بولی & به کمک مباحثی که در بالا عنوان گردید، نمایش داده می‌شوند. یک یال مخصوص برای اتصال گراف‌های عبارات بولی & و عبارات جبری به دیاگرام بسط تیلور تقویت‌شده اضافه می‌کنیم. برای نشان دادن این یال از یک پیکان جهت‌دار استفاده می‌شود. برای اتصال چنین عباراتی، گراف یکی را به کمک یک متغیر میانی در یک گره مخصوص قرار می‌دهیم

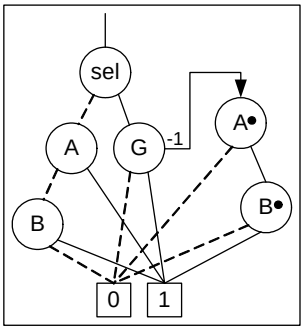
[1] Generalized Shannon

و از آن گره در گراف دیگر استفاده می‌کنیم. به عنوان مثال، برای نمایش رابطه ۷-۵، آنرا به عبارات زیر تبدیل می‌کنیم.

$$sel' \times (A + B) + sel \times G$$
$$G = A \,\&\, B$$

رابطه ۷-۱۲

عبارت اول، یک عبارت جبری بوده و دومی یک عملیات برداری & می‌باشد. نمایش دیاگرام بسط تیلور تقویت‌شده این عبارت‌ها در شکل ۷-٤ نشان داده شده است. از آنجا که sel یک متغیر بیتی می‌باشد، همانطور که در بخش ۷-۵-۱ عنوان گردید؛ از تجزیه شانون برای نمایش این متغیر استفاده می‌شود. متغیرهای A، B و G برداری بوده و بنابراین تجزیه تیلور برای آنها به کار خواهد رفت. پیکان خارج‌شونده از G بیانگر آن است که G یک متغیر میانی بوده و $A \,\&\, B$ را نمایش می‌دهد. مزیت این متغیر میانی هنگامی که اختلاط عملگرها در چندین سطح اتفاق بیفتد، بیشتر آشکار می‌گردد. برای مثال، در رابطه ۷-۱۲ عبارت $(A \,\&\, B) + 1$ را بجای $A \,\&\, B$ در نظر بگیرید. در این حالت، برای نمایش کل عبارت، $A \,\&\, B$ باید در یک گره مخصوص قرار گیرد.



شکل ۷-٤: نمایش دیاگرام بسط تیلور تقویت‌شده رابطه ۷-۱۲

تنها مطلب باقیمانده که باید مورد توجه قرار گیرد قانون دمورگان[1] می‌باشد. یکی از قوانین دمورگان بیان می‌کند که همیشه رابطه بولی برداری زیر برقرار می‌باشد.

$$NOT(A)\ AND\ NOT(B) = NOT(A\ OR\ B)$$

رابطه ۷-۱۳

با استفاده از رابطه ۷-۹، رابطه ۷-۱۰ و رابطه ۷-۱۱ در رابطه ۷-۱۳ و جابجایی متغیرها، رابطه ۷-۱٤ بدست می‌آید.

$$A \,\&\, B = 1 + A + B + (-A - 1) \,\&\, (-B - 1)$$

رابطه ۷-۱٤

[1] De Morgan

رابطه ۷-۱٤ بیان می‌کند که دو روش مختلف برای نمایش عملگر برداری & وجود دارد. دیاگرام بسط تیلور تقویت‌شده باید به گونه‌ای از عملگر & پشتیبانی نماید که دیاگرام حاصل یگانی باشد. برای این منظور، دیاگرام بسط تیلور تقویت‌شده وزن‌های ضرب‌شونده نمایش دو عبارتی که باید با هم & شوند را بررسی می‌کند. این دیاگرام برای نمایش & از روشی (عبارت سمت راست و یا سمت چپ رابطه ۷-۱٤) استفاده می‌کند که عملوندهای عملگر & دارای وزن ضرب‌شونده مثبت باشند. اگر وزن ضرب‌شونده عملوندها منفی باشد، از عبارت سمت راست رابطه ۷-۱٤ استفاده می‌شود. اگر وزن ضرب‌شونده عملوندها مثبت باشد، از عبارت سمت چپ رابطه ۷-۱٤ استفاده می‌شود. نهایتا اگر وزن ضرب‌شونده یکی مثبت و دیگری منفی باشد، برحسب آنکه وزن ضرب‌شونده گره با شناسه بزرگتر مثبت یا منفی باشد، از عبارت سمت چپ یا سمت راست رابطه ۷-۱٤ استفاده خواهد شد. قابل توجه است که شناسه یک مقدار طبیعی یکتا است که در پیاده‌سازی دیاگرام بسط تیلور تقویت‌شده به هر گره نسبت داده می‌شود.

۷-٦ نتایج تجربی

در این بخش، نتایج تجربی حاصل از اجرای الگوریتم‌ها بر روی یک ماشین پنتیوم ٤ با یک گیگابایت حافظه ارائه می‌گردد. همه زمان‌ها برحسب میلی‌ثانیه ارائه شده است. بسته‌های دیاگرام گشتاور دودویی، دیاگرام بسط تیلور و دیاگرام بسط تیلور تقویت‌شده بوسیله Visual C++ نسخه ٦ پیاده‌سازی گردیده‌اند.

در سری اول آزمایش‌ها، نمایش بولی سطح بیت دیاگرام بسط تیلور و دیاگرام بسط تیلور تقویت‌شده مقایسه شده‌اند. محک‌های سطح انتقال‌ثبات که در این آزمایش مورد استفاده قرار گرفته‌اند، به نحوی انتخاب شده‌اند که فقط عملگرهای جبری در مسیرداده آنها وجود داشته باشد. بنابراین اختلاف این دیاگرام‌ها از نمایش بخش‌های بولی سطح بیت ناشی شده است. جدول ۷-۳ خلاصه‌ای از نتایج بدست آمده برای این محک‌ها را ارائه می‌کند. از پنج محک مورد استفاده، Paulin یک حل‌کننده معادله دیفرانسیل بوده و به تفصیل در [61] ارائه گردیده است. محک Chain_mult مداری است که از [62] گرفته شده است. محک‌های SimpleCPU و SimpleRTL در [63] معرفی گردیده‌اند. طول کلمه همه محک‌های این آزمایش ۸ بیت می‌باشد. همه دیاگرام‌ها برحسب ترتیب واحدی میان متغیرها ساخته شده‌اند.

همانطور که جدول ۷-۳ نشان می‌دهد، دیاگرام بسط تیلور تقویت‌شده از نظر تعداد گره و زمان تبدیل بهتر از دیاگرام بسط تیلور عمل می‌نماید. نمایش سطح بردار عملگرهای جبری در دیاگرام بسط تیلور تقویت‌شده و دیاگرام بسط تیلور یکسان می‌باشد. بنابراین مزیت‌های بدست آمده در این آزمایش بدلیل نمایش منطقی سطح بیت دیاگرام بسط تیلور تقویت‌شده بوده که شبیه دیاگرام تصمیم‌گیری دودویی می‌باشد و از دیاگرام بسط تیلور

فشرده‌تر است. به دلیل تعداد گره‌های کمتر، زمان تبدیل دیاگرام بسط تیلور تقویت‌شده کمتر می‌باشد. همچنین، دیاگرام بسط تیلور از نظر تعداد گره و زمان تبدیل بهتر از دیاگرام گشتاور دودویی می‌باشد. این امر بدلیل نمایش سطح بردار دیاگرام بسط تیلور می‌باشد؛ چرا که نمایش سطح بیت دیاگرام بسط تیلور و دیاگرام گشتاور دودویی مشابه می‌باشد.

جدول ۷-۳: مقایسه ساختمان‌داده‌های مختلف بوسیله محک‌های انتقال‌ثبات ساده

	BMD		**TED**		**ETED**	
	Nodes	Time	Nodes	Time	Nodes	Time
SimpleRTL	774	16	159	6	94	5
Paulin	1017	109	380	16	208	9
Chain_mult	282	33	114	12	89	7
SimpleCPU	687	46	407	15	249	10
Avenhaus	1093	50	372	15	211	9

در سری دوم آزمایش‌ها، همه قابلیت‌های دیاگرام بسط تیلور تقویت‌شده در نمایش طرح‌های انتقال‌ثبات با دیاگرام بسط تیلور و دیاگرام گشتاور دودویی مقایسه شده است. جدول ۷-٤ خلاصه‌ای از نتایج بدست آمده برای مدارهای محک را نشان می‌دهد. محک Tseng با جزئیات دقیق در [64] توصیف شده است. محک ASPP4 از مرجع [65] گرفته شده و رفتار Paulin یا Tseng را شبیه‌سازی می‌کند. این دو محک طول کلمه‌ای برابر ۸ بیت دارند. محک Parwan یک پردازنده ۸ بیتی بوده و Sayeh یک پردازنده ١٦ بیتی می‌باشد.

جدول ۷-٤: مقایسه ساختمان‌داده‌های مختلف بوسیله محک‌های انتقال‌ثبات واقعی

	BMD		**TED**		**ETED**	
	Nodes	Time	Nodes	Time	Nodes	Time
Tseng	771	30	771	31	170	1
ASPP4	3297	1130	3297	1132	612	37
Parwan	5321	1500	5321	1501	724	40
Sayeh	X	X	X	X	731	42

آنچنان که از جدول ۷-٤ مشهود است، بدلیل نمایش بولی سطح بردار، دیاگرام بسط تیلور تقویت‌شده بهتر از دیاگرام بسط تیلور و دیاگرام گشتاور دودویی می‌باشد. محک Parwan که یک پردازنده ۸ بیتی است در زمان قابل قبول به دیاگرام گشتاور دودویی، دیاگرام بسط تیلور و دیاگرام بسط تیلور تقویت‌شده تبدیل شده است. از طرف دیگر، نمایش دیاگرام گشتاور دودویی و دیاگرام بسط تیلور پردازنده Sayeh در مدت ٦٠ ثانیه ساخته

نشد؛ اما نمایش دیاگرام بسط تیلور تقویت‌شده این پردازنده در مدت زمانی برابر با مدت زمان تبدیل Parwan، حدود ٤٠ میلی‌ثانیه، ساخته شد. لازم به ذکر است که پیچیدگی مسیرداده پردازنده‌های Sayeh و Parwan تقریبا برابر می‌باشد و تنها طول کلمه آن دو با هم متفاوت است.

تعداد گره‌ها و زمان تبدیل دیاگرام بسط تیلور تقویت‌شده مستقل از طول کلمه می‌باشد؛ اما تعداد گره‌ها و زمان تبدیل در دیاگرام گشتاور دودویی و دیاگرام بسط تیلور به طول کلمه محک‌ها وابسته می‌باشد.

در بسیاری از مسیرداده‌های معمولی همچنان که جدول ۷-٤ نشان می دهد، دیاگرام بسط تیلور مزیتی بر دیاگرام گشتاور دودویی ندارد. از طرف دیگر، دیاگرام بسط تیلور تقویت‌شده برتری قابل توجهی نسبت به دیاگرام گشتاور دودویی و دیاگرام بسط تیلور دارد.

۷-۷ نتیجه‌گیری

در این فصل دیاگرام بسط تیلور تقویت‌شده ارائه گردید. این دیاگرام در نمایش عبارات بولی سطح بیت مشابه دیاگرام تصمیم‌گیری دودویی می‌باشد. همچنین در نمایش عبارات جبری سطح‌کلمه مشابه دیاگرام بسط تیلور می‌باشد. از طرف دیگر، دیاگرام بسط تیلور تقویت‌شده می‌تواند عملگرهای بولی سطح‌کلمه (بردار) را نمایش دهد؛ بنابراین لازم نیست که این عملگرها را به معادل بیتی‌شان تبدیل کرد.

دیاگرام بسط تیلور تقویت‌شده نمایشی بهینه برای طرح‌های انتقال‌ثبات می‌باشد. این دیاگرام قادر است تعداد زیادی از طرح‌های انتقال‌ثبات را به شکل فشرده نمایش دهد.

فصل هشتم: دست‌آوردها و کارهای آینده

هدف از این پایان‌نامه فراهم آوردن بخشی از زیرساخت‌های لازم برای ارزیابی‌رسمی مدارهای دیجیتال در سطح انتقال‌ثبات می‌باشد. یکی از مهمترین لازمه‌های ارزیابی‌رسمی در سطح انتقال‌ثبات، توانایی نمایش یگانی توصیفات انتقال‌ثبات بوسیله یک ساختمان‌داده بهینه است. این ساختمان‌داده می‌تواند همان موفقیت‌هایی را که دیاگرام تصمیم‌گیری دودویی برای ارزیابی‌رسمی در سطح گیت به ارمغان آورد، برای ارزیابی‌رسمی در سطح انتقال‌ثبات به ارمغان آورد. بعد از بررسی ساختمان‌داده‌های موجود، دیاگرام بسط تیلور به عنوان زیرساخت ساختمان‌داده جدید برگزیده شد. با اعمال تغییرات و اضافه‌کردن قابلیت‌هایی به دیاگرام بسط تیلور، ساختمان‌داده مناسبی برای سطح انتقال‌ثبات پیشنهاد شد. مهمترین دست‌آوردهای این پایان‌نامه عبارتند از:

- ارائه یک پیاده‌سازی بهینه برای دیاگرام بسط تیلور
- اعمال تغییراتی برای بهبود نمایش عملگرهای بولی سطح بیت به دیاگرام بسط تیلور
- اضافه کردن قابلیت نمایش عملگرهای بولی سطح‌کلمه (بردار) به دیاگرام بسط تیلور

اگرچه در این پایان‌نامه قدم‌های چندی برداشته شده است تا به هدف نهایی که همانا داشتن موتورهای ارزیابی‌رسمی در سطح انتقال‌ثبات می‌باشد، نزدیک شویم؛ اما برای رسیدن به این هدف لازم است قدم‌های بیشتری برداشته شود. این قدم‌ها عبارتند از:

- اضافه کردن قابلیت نمایش عملگرهای مقایسه‌ای سطح‌کلمه (بردار) به دیاگرام بسط تیلور
- ارائه الگوریتم‌هایی برای بررسی خصوصیت در سطح انتقال‌ثبات به کمک ساختمان‌داده حاصل
- پیاده‌سازی موتور ارزیابی‌رسمی در سطح انتقال‌ثبات

مراجع

[1] F. Bacchini, R. Damiano, B. Bentley, K. Baty, K. Normoyle, M. Ishii, and E. Yogev, "Verification: What works and what doesn't", in Proc. of Design Automation Conference, pp. 274-274, June, 2004.

[2] D. L. Dill, "What's between simulation and formal verification?", in Proc. of Design Automation Conference, pp. 328–329, June, 1998.

[3] C. Kern, and M. R. Greenstreet, "Formal verification in hardware design: A survey", ACM Transactions on Design Automation of Electronic Systems, Vol. 4, No. 2, pp. 123-193, April, 1999.

[4] S.-Y. Huang, and K.-T. Cheng, Formal equivalence checking and design debugging, Kluwer Academic Publishers, Norwell, MA, USA, 1998.

[5] A. Kuehlmann, and F. Krohm, "Equivalence checking using cuts and heaps", in Proc. of Design Automation Conference, pp. 263-268, June, 1997.

[6] J. R. Burch, E. M. Clarke, D. E. Long, K. L. McMillan, and D. L. Dill, "Symbolic model checking for sequential circuit verification", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 13, No. 4, pp. 401-424, April, 1994.

[7] E. M. Clarke, O. Grumberg, and D. Peled, Model checking, MIT Press, Cambridge, Massachusetts, USA, 1999.

[8] S. B. Akers, "Binary decision diagrams", IEEE Transactions on Computers, Vol. c-27, No. 6, pp. 509-516, June, 1978.

[9] P. Ashar, S. Devadas, and A. Ghosh, "Boolean satisfiability and equivalence checking using general binary decision diagrams", in Proc. of International Conference on Computer Design, pp. 259-264, October, 1991.

[10] R. I. Bahar, E. A. Frohm, C. M. Gaona, G. D. Hachtel, E. Macii, A. Pardo, and F. Somenzi, "Algebraic decision diagrams and their applications", in Proc. of International Conference on Computer-Aided Design, pp. 188-191, November, 1993.

[11] J. Bern, C. Meinel, and A. Slobodova, "Efficient OBDD-based Boolean manipulation in CAD beyond current limits", in Proc. of Design Automation Conference, Pages 408-413, June, 1995.

[12] K. S. Brace, R. L. Rudell, and R. E. Bryant, "Efficient implementation of a BDD package", in Proc. of Design Automation Conference, pp. 40-45, June, 1990.

[13] R. E. Bryant, "Graph-based algorithms for Boolean function manipulation", IEEE Transactions on Computers, Vol. c-35, No. 8, pp. 677-691, August, 1986.

[14] R. E. Bryant, "On the complexity of VLSI Implementations and graph representations of Boolean functions with application to integer multiplication", IEEE Transactions on Computers, Vol. 40, No. 2, pp. 205-213, February, 1991.

[15] R. E. Bryant, "Symbolic Boolean manipulation with ordered binary decision diagrams", ACM Computing Surveys, Vol. 24, No. 3, pp. 293-318, September, 1992.

[16] R. E. Bryant, and Y.-A. Chen, "Verification of arithmetic circuits with binary moment diagrams", in Proc. of Design Automation Conference, pp. 535-541, June, 1995.

[17] E. Clarke, K. L. McMillan, X. Zhao, M. Fujita, and J. C.-Y. Yang, "Spectral transforms for large Boolean functions with application to technology mapping", in Proc. of Design Automation Conference, pp. 54-60, June, 1993.
[18] R. Drechsler, A. Sarabi, M. Theobald, B. Becker, and M. A. Perkowski, "Efficient representation and manipulation of switching functions based on ordered kronecker functional decision diagrams", in Proc. of Design Automation Conference, pp. 415-419, June, 1994.
[19] M. Fujita, H. Fujisawa, and N. Kawato, "Evaluation and improvement of Boolean comparison method based on binary decision diagrams", in Proc. of International Conference on Computer-Aided Design, pp. 2-5, November, 1988.
[20] J. Gergov, and C. Meinel, "Efficient Boolean manipulation with OBDD's can be extended to FBDD's", IEEE Transactions on Computers, Vol. 43, No. 10, pp. 1197-1209, October, 1994.
[21] U. Kebschull, E. Schubert, and W. Rosentiel, "Multilevel logic synthesis based on functional decision diagrams", in Proc. of European Design Automation Conference, pp. 43-47, March, 1992.
[22] Y.-T. Lai, and S. Sastry, "Edge-valued binary decision diagrams for multi-level hierarchical verification", in Proc. of Design Automation Conference, pp. 608-613, June, 1992.
[23] Y.-T. Lai, M. Pedram, and S. B. K. Vrudhula, "EVBDD-based algorithms for integer linear programming, spectral transformation, and function decomposition", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 13, No. 8, pp. 959-975, August, 1994.
[24] S. Malik, A. Wang, R. K. Brayton, and A. Sangiovanni-Vincentelli, "Logic verification using binary decision diagrams in a logic synthesis environment", in Proc. of International Conference on Computer-Aided Design, pp. 6-9, November, 1988.
[25] S.-i. Minato, "Zero-suppressed BDDs for set manipulation in combinatorial problems", in Proc. of Design Automation Conference, pp. 272-277, June, 1993.
[26] S.-i. Minato, Binary Decision Diagrams and Applications for VLSI CAD, Kluwer Academic Publishers, Norwell, MA, USA, 1995.
[27] H. Ochi, K. Yasuoka, and S. Yajima, "Breadth-first manipulation of very large binary-decision diagrams", in Proc. of International Conference on Computer-Aided Design, pp. 48-55, November, 1993.
[28] S. Ponzio, "A lower bound for integer manipulation with read-once only branching programs", in Proc. of ACM Symposium on Theory of Computing, pp. 130-139, May, 1995.
[29] O. Schroer, and I. Wegener, "The theory of zero-suppressed BDDs and the number of knight's tours", Formal Methods in System Design, Vol. 13, No. 3, pp. 235-253, November, 1998.
[30] D. Sieling, and I. Wegener, "Graph driven BDDs—a new data structure for Boolean functions", Theoretical Computer Science, Vol. 141, No. 1&2, pp. 283-310, April, 1995.
[31] R. Rudell, "Dynamic variable ordering for ordered binary decision diagrams", in Proc. of International Conference on Computer-Aided Design, pp. 42-47, November, 1993.
[32] M. Ciesielski, P. Kalla, Z. Zheng, and B. Rouzeyre, "Taylor expansion diagrams: a new representation for RTL verification", in Proc. of High-Level Design Validation and Test Workshop, pp. 70-75, November, 2001.

[33] M. J. Ciesielski, P. Kalla, Z. Zheng, and B. Rouzeyre, "Taylor expansion diagrams: a compact, canonical representation with applications to symbolic verification", in Proc. of Design, Automation and Test in Europe, pp. 285-289, March, 2002.
[34] P. Kalla, M. Ciesielski, E. Boutillon, and E. Martin, "High-level design verification using Taylor expansion diagrams: first results", in Proc. of High-Level Design Validation and Test Workshop, pp. 13-17, October, 2002.
[35] R. K. Brayton, G. D. Hachtel, A. Sangiovanni-Vincentelli, F. Somenzi, A. Aziz, S.-T. Cheng, S. Edwards, S. Khatri, Y. Kukimoto, A. Pardo, S. Qadeer, R. Ranjan, S. Sarwary, G. Shiple, S. Swamy, and T. Villa, "VIS: a system for verification and synthesis", in Proc. of International Conference on Computer-Aided Verification, pp. 428-432, July, 1996.
[36] K. L. McMillan, Synthesis Model Checking, Kluwer Academic Publishers, Norwell, MA, USA, 1993.
[37] D. Cyrluk, O. Moller, and H. Ruess, "An efficient decision procedure for the theory of fixed-size bit vectors", in Proc. of International Conference on Computer-Aided Verification, pp. 60-71, June, 1997.
[38] C. W. Barlett, D. L. Dill, and J. R. Levitt, "A decision procedure for bit-vector arithmetic", in Proc. of Design Automation Conference, pp. 522-527, June, 1998.
[39] H. B. Enderton, A mathematical introduction to logic, Academic Press, New York, USA, 1972.
[40] T. Bultan, R. Gerber, and C. League, "Verifying systems with integer constraints and Boolean predicates: a composite approach", in Proc. of International Symposium on Software testing and analysis, pp. 113-123, March, 1998.
[41] S. Devadas, K. Keutzer, and A. Krishnakumar, "Design verification and reachability analysis using algebraic manipulation", in Proc. of International Conference on Computer Design, pp. 250-258, October, 1991.
[42] Z. Zhou, and W. Burleson, "Equivalence checking of datapaths based on canonical arithmetic expressions", in Proc. of Design Automation Conference, pp. 546-551, June, 1995.
[43] S. Horeth, and R. Drechsler, "Formal verification of word-level specifications", in Proc. of Design Automation and Test in Europe, pp. 52-58, March, 1999.
[44] R. Drechsler, and B. Becker, "Ordered kronecker functional decision diagrams—a data structure for representation and manipulation of Boolean functions", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 17, No. 10, pp. 965-973, October, 1998.
[45] E. M. Clarke, M. Fujita, and X. Zhao, "Hybrid decision diagrams. Overcoming the limitations of MTBDDs and BMDs", in Proc. of International Conference on Computer-Aided Design, pp. 159-163, November, 1995.
[46] R. Drechsler, B. Becker, and S. Ruppertz, "The K*BMD: a verification data structure", Design & Test of Computers, Vol. 14 , No. 2 , pp. 51-59, April-June, 1997.
[47] R. Drechsler, B. Becker, and S. Ruppertz, "K*BMD: a new data Structure for Verification", in Proc. of European Design and Test Conference, pp. 2-8, March, 1996.
[48] R. Drechsler, B. Becker, and S. Ruppertz, "Manipulation algorithms for K*BMDs", in Proc. of Tools and Algorithms for the Construction and Analysis of Systems, pp. 4-18, April, 1997.
[49] Y. Parasuram, E. Stabler, and S.-K. Chin "Parallel implementation of BDD algorithms using a distributed shared memory", in Proc. of Hawaii International Conference on System Sciences, Pages 16-25, Januray 1994.

[50] T. Stornetta, and F. Brewer, "Implementation of an efficient parallel BDD package", in Proc. of Design Automation Conference, pp. 641-644, June, 1996.
[51] B. Becker, R. Drechsler, and R. Werchner, "On the relation between BDDs and FDDs", Information Computing, Vol. 123, No. 2, pp. 185-197, December, 1995.
[52] R. Drechsler, M. Theobald, and B. Becker, "Fast OFDD based minimization of fixed polarity Reed-Muller expressions", IEEE Transactions on Computers, Vol. 45 , No. 11, pp. 1294-1299, November, 1996.
[53] R. Drechsler, and B. Becker, "Sympathy: Fast exact minimization of fixed polarity Reed-Muller expressions for symmetric functions", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 16, No. 1, pp. 1-5, Januray, 1997.
[54] Y-T. Lai, M. Pedram, and S. B. Vrudhula, "FGILP: an integer linear program solver based on function graphs", in Proc. of International Conference on Computer-Aided Design, pp. 685-689, November, 1993.
[55] S. Minato, N. Ishiura, and S. Yajima, "Shared binary decision diagram with attributed edges for efficient Boolean function manipulation", in Proc. of Design Automation Conference, pp. 52-57, Januray, 1991.
[56] F. Somenzi, *CUDD: CU Decision Diagram Package*, Release 2.3.0., University of Colorado at Boulder, 1998.
[57] F. Brglez, and H. Fujiwara, "A neutral netlist of 10 combinational circuits and a target translator in fortran", in Proc. of International Symposium on Circuits and Systems, pp. 663-698, June, 1985.
[58] Y.-T. Lai, M. Pedram, and S. B. K. Vrudhula, "Formal verification using edge-valued binary decision diagrams", IEEE Transactions on Computers, Vol. 45, No. 2, pp. 247-255, February, 1996.
[59] B. Becker, R. Drechsler, and R. Enders, "On the representational power of bit-level and word-level decision diagrams", in Proc. of Asia and South Pacific Design Automation Conference, pp. 461-467, January, 1997.
[60] G. Fey, R. Drechsler, and M. Ciesielski, "Algorithms for Taylor expansion diagrams", in Proc. of International Symposium on Multiple-Valued Logic, pp. 235-240, May, 2002.
[61] I. Ghosh, A. Raghunathan, and N. K. Jha, "A design-for-testability technique for register-transfer level circuits using control/data flow extraction", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 17, No. 8, pp. 706-723, August, 1998.
[62] S. Ravi, I. Ghosh, R. K. Roy, and S. Dey, "Controller resynthesis for testability enhancement of RTL controller/data path circuits", in Proc. of International Conference on VLSI Design, pp. 193-198, January, 1998.
[63] S. Ravi, G. Lakshminarayana, and N. K. Jha, "TAO: regular expression-based register-transfer level testability analysis and optimization", IEEE Transactions on Very Large Scale Integration Systems, Vol. 9, No. 6, pp. 824-832, December, 2001.
[64] I. Ghosh, N. K. Jha, and S. Bhawmik, "A BIST scheme for RTL controller-data paths based on symbolic testability analysis", in Proc. of Design Automation Conference, pp. 554-559, June, 1998.
[65] I. Ghosh, A. Raghunathan, and N. K. Jha, "Hierarchical test generation and design for testability methods for ASPPs and ASIPs", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 18, No. 3, pp. 357-370, March, 1999.

مقالات استخراج‌شده از پایان‌نامه

Abstract

Today's leading chip and system companies are faced with ever increasing design verification challenges; industrial studies reveal that as much as 50% of the total schedule is being spent in verfication. Large companies, with almost infinite resources, have shown that throwing CPU cycle and people at the simulation problem still does not guarantee a level of coverage desired by the design team.

The complexity of today's designs has grown to a point were verification has become a major bottleneck. With IC implementation there is sign-off and you clearly know when you are done. With verification the challenge lies in the fact that while the goal is clean designs and predictable schedules, today's design verification methodology is empirical, therefore a company can never really be sure when it is truly done verifying its design. Instead, you must continue to run various simulations until you decide that you have done enough. This is not because you are fully confident that the design is bug-free, but because you have simply run out of time.

Formal verification is a reliable method that was proposed and improved in the last two decades and is a suitable complement for conventional simulation. Formal verification solves the problems of simulation by applying mathematical rules for proving the validation of a design implementation.

Increasing size and complexity of digital designs has made essential the need to address critical formal verification issues at the early stages of design cycle. This requires robust, automated formal verification tools at higher (RT or behavioral) levels of abstraction. There already are tools for formal verification of designs implemented at lower levels of abstraction, such as gate-level and circuit/layout level. However tools that can verify a high-level specification have not yet reached the desired level of automation and maturity. This is partly due to the lack of a good graph-based representation for the RT-level descriptions. While for the gate level designs, BDD and its variations offer good solutions for their representation, most high-level verification tools synthesize such descriptions to be able to take advantage of BDDs that are at the gate-level.

The aim of this thesis is to provide a high-level graph-based representation for manipulation of RT-level descriptions. To achieve this, we have used Taylor Expansion Diagrams (TEDs). TED has much superiority over other graph-based representations in manipulation of RTL designs, but it has some limitations. We enhance TED to overcome its shortcomings to achieve a suitable data structure for representing RT-level designs.

University of Tehran

Faculty of Engineering

Department of Electrical and Computer Engineering

Thesis Title:

An Efficient Data Structure for RTL Representation

By:

Pejman Lotfi-Kamran

Advisor:

Dr. Zainalabedin Navabi

Co-Advisor:

Dr. Mehran Massoumi

A thesis submitted to the Graduate Studies Office in partial fulfillment of the requirements for the degree of Master of Science in Computer Engineering

February 2005