طراحی شبکه‌ی پایین بر بهینه DCVS با استفاده از مفهوم توابع هم ارز

حسین محتشمی*، کیوان ناوی†، علیرضا مهینی‡

چکیده

در این مقاله سعی شده است تا روشی با الگوریتم مشخص جهت یافتن مدار DCVS بهینه برای توابع منطقی ارائه گردد. برای نیل به این منظور ابتدا با بررسی خانواده‌های مختلف مدارهای منطقی، بخصوص مدارهای DCVS، تعریفی برای یک مدار DCVS بهینه ارائه گردیده است. سپس با بررسی روش‌های مختلف موجود جهت ایجاد ساختار متناظر هر تابع منطقی در مدار معادل DCVS، روشی پیشنهاد گردیده است که توسط آن می‌توان به ساختارهایی با سرعت بیشتر و مصرف توان پایین تر و نیز هزینه‌ی پیاده سازی کم‌تری دست یافت. مشخصه‌ی اصلی این روش آن است که بر خلاف روش‌های موجود، طراح درگیر جزئیات طراحی نشده و بدون نیاز به قوه خلاقیت و ابتکار، می‌تواند مدار بهینه‌ی معادل یک تابع منطقی را تولید نماید.

کلمات کلیدی

DCVS، مدارهای مجتمع بسیار فشرده، شبکه‌ی پایین بر، توابع هم ارز و دیاگرام تصمیم دودویی.

Pull Down Network Minimization of DCVS based on Equivalent Functions concept

Hossein Mohtashami, Keivan Navi, Alireza Mahini

Abstract

In this paper, a novel method with well defined algorithm is applied for constructing optimized DCVS circuits for a Boolean function. In this approach a definition for optimized DCVS circuits, in specific DCVS logic families is presented by surveying different logic families; afterwards a method which can produce fast, low power, low cost DCVS structures from random Boolean functions is demonstrated. The distinction of this approach despite the others is that it is not heuristic; the designer is not interfere with details and also with no intelligence can create the optimized DCVS circuit from a defined Boolean function.

Keywords

DCVS(Differential Cascode Voltage Switch Logic), VLSI circuit, Pull Down Network (PDN), Equivalent Functions, Binary Decision Diagram (BDD).

* کارشناس ارشد مهندسی کامپیوتر در گرایش معماری سیستمهای کامپیوتری mohtashami@excite.com.
† استادیار دانشکده‌ی مهندسی برق و کامپیوتر دانشگاه شهید بهشتی navi@sbu.ac.ir.
‡ کارشناس ارشد مهندسی کامپیوتر در گرایش معماری سیستمهای کامپیوتری mahini@comp.iust.ac.ir.

۱- مقدمه

با رشد روز افزون تکنولوژی‌های ساخت مدارهای مجتمع و ورود به عرصه مدارهای بسیار پرتراکم ULSI ، وابستگی طراحان مدار به ابزارهای طراحی خودکار بیشتر می‌گردد. تا چند دهه گذشته ارائه توصیف سطح بالا و دریافت یک طراحی بهینه در سطح ترانزیستور تنها به عنوان یک هدف مطرح بود و امروزه این امر، تبدیل به یک وسیله گردیده است. طراحی مدارهای مجتمعی که دارای دهها میلیون ترانزیستور می‌باشد بدون استفاده از ابزارهای CAD امکان پذیر نبوده و در این بین در اختیار داشتن ابزارهای قوی CAD به منزله کم شدن اشتباهات طراحی و پایین آمدن هزینه‌های ساخت و کوتاه‌تر شدن زمان طراحی می‌باشد. ابزارهای CAD یک توصیف کامل و جامع سطح بالا را از طراح گرفته و با استفاده از روش‌ها و الگوریتم‌هایی پیاده‌سازی لازم را در پایین‌ترین سطح به نحوی انجام می‌دهد که نیازمندی‌های طراح تامین گردد. در این بین مجالی برای روش‌هایی که بصورت ابتکاری یا دستی قادر به حل مشکل هستند، وجود ندارد و تنها روش-های مبتنی بر الگوریتم‌های مشخص و پایان پذیر و البته بهینه در این رابطه قابل استفاده می‌باشند.

درگذشته‌ی نه چندان دور طراحی مدار جهت پیاده سازی توابع منطقی در سطح ترانزیستور، از مواردی بود که طراح بطور مستقیم با آن روبرو بود و به همین دلیل تمام روش‌های طراحی و پیاده‌سازی مدارها بصورت دستی و گاهی هم ابتکاری انجام می‌گردید. امروزه با استفاده از ابزارهای CAD با کتابخانه های مختلف، الگوریتم‌های دقیق و جامع جایگزین روش‌های دستی و ابتکاری شده‌اند. در این نوشتار روشی بهینه جهت پیاده‌سازی توابع منطقی در خانواده‌های مدارهای DCVS ارائه گردیده است که مشخصه‌ی اصلی آن داشتن الگوریتمی جامع و بهینه می‌باشد که مناسب جهت استفاده در ابزارهای CAD می‌باشد.

۲- روشهای موجود طراحی شبکه ی پایین بر DCVS

در طراحی مدارهای DCVS مهمترین چالش، طراحی شبکه‌ی پایین بر متناظر با تابع منطقی دلخواه است. لازم به توضیح است که بین توابع منطقی و شبکه‌ی پایین بر مدارهای DCVS تناظر یک به یکی وجود ندارد و برای یک تابع منطقی می‌توان شبکه‌های پایین بر مختلفی را طراحی نمود. ساختار شبکه‌ی پایین بر در DCVS به نحوی است که معمولاً دارای قسمت های مشترکی بوده و با ادغام این اشتراکات می‌توان شبکه‌ی پایین بر ساده‌تری را برای یک تابع منطقی بدست آورد. در بین مجموعه شبکه‌های پایین بر نظیر یک تابع منطقی، آن که کمترین تعداد ترانزیستور را نیاز داشته باشد دارای کمترین هزینه پیاده سازی، ظرفیت خازن ورودی، مصرف توان و تاخیر می‌باشد. یافتن یک طراحی بهینه برای توابع با تعداد ورودی‌های کم شاید حتی بصورت دستی و روش آزمایش و خطا امکان پذیر باشد ولی در صورتیکه که تعداد متغییرها تنها اندکی افزایش یابد، یافتن یک طراحی بهینه با استفاده از روش‌های آزمایش و خطا امکان پذیر نبوده و نیاز به استفاده از روش‌های اصولی تر محرز می‌باشد.

روش‌های موجود برای طراحی شبکه‌ی پایین بر در مدارهای DCVS را می‌توان بر اساس موارد زیر با یکدیگر مقایسه نمود:

داشتن الگوریتم مشخص: آیا روش دارای الگوریتم مشخص و کاملی است یا ابتکاری می‌باشد؟

پایان پذیر بودن: آیا روش مورد بحث دارای نقطه پایان مشخصی می-باشد یا خیر؟

تولید شبکه‌ی پایین بر بهینه: روش مورد بحث تا چه حد قادر به یافتن بهترین شبکه‌ی پایین بر می‌باشد؟

لازم به توضیح است که تعداد زیادی از خانواده های مدارها منطقی از شبکه ی پایین بر DCVS استفاده می نمایند (مرجع [1]). به عنوان مثال خانواده های دامینوی دو خطی، SDSL، ECDL، DCSL[1] و بسیاری دیگر از خانواده های مدارها منطقی دارای شبکه ی پایین بر مطابق منطق DCVS هستند. در نتیجه یافتن روشی جهت تولید شبکه ی پایین بر بهینه در خانواده‌های مدارهای منطقی مختلفی کاربرد دارد. در ادامه روش‌هایی که تا کنون برای تولید شبکه‌ی پایین-بر مدارهای DCVS مطرح شده است؛ توضیح داده می‌شود.

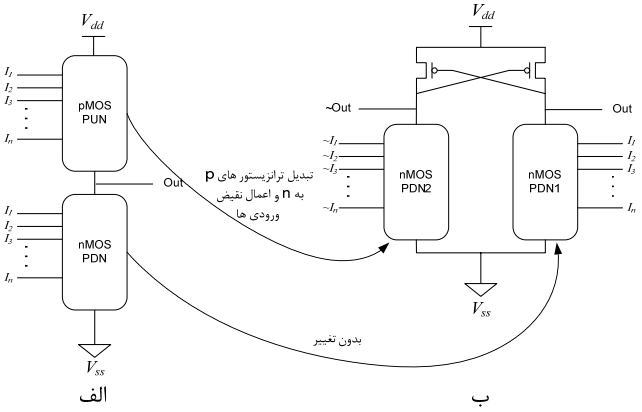
۲-۱- ایجاد شبکه ی پایین بر بر اساس ساختار منطق CMOS ایستا

با اعمال پاره‌ای تغییرات بر روی شبکه‌های پایین بر و بالابر در پیاده-سازی CMOS یک تابع منطقی، ساختار شبکه‌ی پایین بر مدار DCVS تابع، بدست خواهد آمد[2]. در شکل(۱) نحوه تبدیل یک مدار CMOS به مدار معادل آن در منطق DCVS نشان داده شده است.

این روش دارای الگوریتم مشخص و پایان پذیر می‌باشد ولی نمی‌تواند بهترین ساختار شبکه‌ی پایین‌بر نظیر هر تابع منطقی را ارائه دهد و برای هر تابع منطقی، مدار DCVS آن نسبت به مدار CMOS آن دو ترانزیستور بیشتر مصرف می‌نماید. البته این مدار DCVS هنوز هم به علت عدم استفاده از ترانزیستورهای PMOS در تولید تابع منطقی نسبت به مدار CMOS برتری دارد. در ادامه روش‌هایی که ساختار شبکه‌ی پایین بر در آنها ساده‌تر شده است، بررسی خواهند شد.

۲-۲- طراحی شبکه‌ی پایین بر با استفاده از جدول کارنو

با استفاده از جدول کارنو می‌توان برای توابعی با تعداد متغییر کمتر از شش، شبکه‌ی پایین بر را برای مدارهای DCVS طراحی نمود[3]. این روش برای تعداد متغییر کم نسبتاً ساده بوده و از دسته روش‌های ابتکاری می‌باشد.

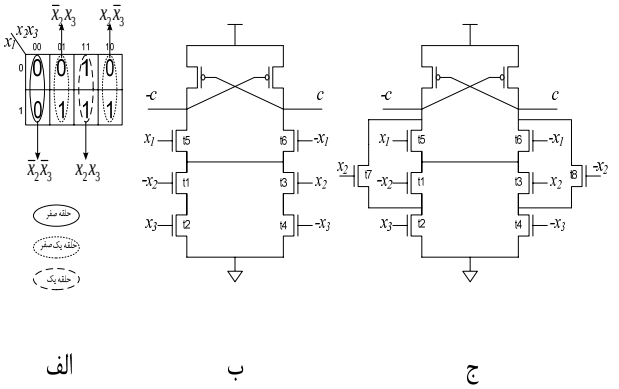


شکل(۱): استفاده از درخت های پایین بر و بالابر CMOS برای تولید مدار DCVS.

در این روش ابتدا باید جدول کارنو برای تابع مفروض کشیده شود. سپس خانه‌های جدول توسط حلقه‌های صفر، یک، صفریک و یک صفر دسته بندی می‌گردد به نحوی که تمام خانه‌های جدول حداقل توسط یکی از انواع حلقه‌ها پوشیده گردد. در انتخاب دسته ها باید توجه شود که هر چه دسته‌بندی‌های انجام شده تعداد خانه‌ی بیشتری داشته باشد در نهایت ساختار شبکه‌ی پایین بر بدست آمده ساده تر می‌باشد. در شکل(۲) نحوه‌ی انتخاب دسته بندی فوق برای یک تابع محاسبه رقم نقلی آمده است.

مشخص است که ساختار درخت بدست آمده برای یک تابع مفروض منحصر به فرد نبوده و با توجه به نحوه دسته‌بندی خانه های جدول کارنو و همچنین تبدیل دسته بندی‌های انجام شده در شبکه‌ی پایین-بر، تعداد زیادی شبکه‌ی پایین بر برای هر تابع منطقی می‌توان ایجاد نمود.

اشکال عمده‌ی این روش آن است که دارای الگوریتم مشخصی نبوده و یک روش ابتکاری می‌باشد. به همین دلیل مشخص نیست که آیا پیاده‌سازی انجام شده، بهترین پیاده‌سازی ممکن می‌باشد یا خیر. این روش در حالت‌هایی که تعداد متغییرها زیاد باشد زمان بر بوده و برای تعداد بیشتر از شش متغییر قابل استفاده نمی باشد.

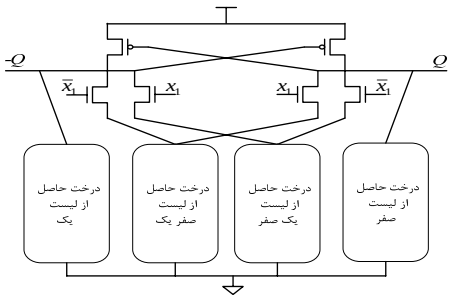


شکل(۲): ایجاد درخت پایین بر تابع محاسبه رقم نقلی توسط جدول کارنو.

۲-۳- طراحی شبکه‌ی پایین بر مبتنی بر روش کویین مک کلاسکی[۲]

این روش در حقیقت تعمیمی بر روش طراحی شبکه‌ی پایین بر با استفاده از جدول کارنو می‌باشد و در آن محدودیتی بر روی تعداد متغییرها وجود ندارد[3]. در این روش حلقه‌های صفر، یک، یک صفر و صفر یک در جدول‌هایی نوشته شده و سپس با استفاده از روش کویین مک کلاسکی تمام *ایجاب کننده های نخستین* [۳] (PI)بدست می‌آید و سپس یک پوشش کامل از آنها انتخاب شده و بر اساس آن شبکه‌ی پایین بر طراحی می‌گردد. ساختار نهایی شبکه‌ی پایین بر در این روش در شکل(۳) آمده است.

در این روش جدول صفر و جدول یک با استفاده از ورودی‌های صفر و ورودی‌های یک تابع منطقی داده شده، ایجاد می‌شود و از روی آن‌ها جداول یک صفر و صفر یک تهیه شده و سپس توسط روش کویین مک کلاسکی، PIهای هر جدول مشخص شده و در نهایت یک پوشش کامل از آنها بدست می‌آید که با استفاده از آن درخت‌های موجود در شکل(۳) تهیه می‌گردد.



شکل(۳): ساختار کلی شبکه‌ی پایین بر مبتنی بر روش کویین مک کلاسکی.

مرحله ایجاد ساختار درخت های پایین بر با استفاده از PI های بدست آمده ابتکاری بوده و دارای الگوریتم خاصی نمی‌باشد ضمن آنکه در [4] اثبات گردیده است که روش فوق نمی‌تواند بهترین شبکه‌ی پایین‌بر را تولید نماید.

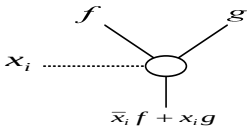
۲-۴- طراحی شبکه‌ی پایین بر با استفاده از OBDD

این روش دارای الگوریتم مشخص بوده و با مدل نمودن توابع منطقی به یک OBDD[۴]، یک پیاده سازی مناسب برای شبکه‌ی پایین بر مدارهای DCVS ارائه می‌دهد[4].

BDD [۵]یک گراف جهت دار بدون حلقه می‌باشد که می‌توان توسط آن توابع منطقی را نمایش داد و با آنها کار نمود[5]. OBDD یک BDD است بطوریکه که در هر مسیر جهت دار آن هر *متغییر تصمیم‌گیری*[۶] حداکثر یکبار و با ترتیب مشخص وجود دارد. متغییرهای تصمیم‌گیری در حقیقت همان ورودی‌های تابع منطقی نمایش داده شده توسط BDD هستند و نحوه عملکرد BDD را تعیین می‌نمایند.

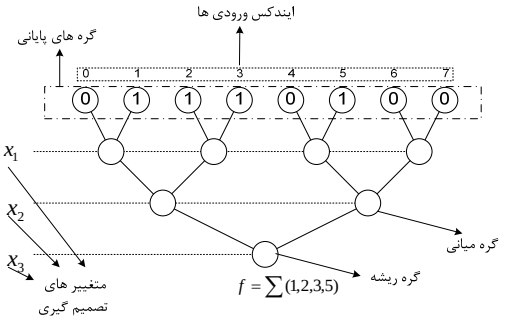
یک OBDD شامل تعدادی گره تصمیم گیری و تعدادی گره پایانی است که توسط یال‌هایی به یکدیگر متصل شده‌اند. به هر گره دو یال

وارد می‌شود و بجز گره ریشه هرگره تصمیم‌گیری یک خروجی دارد. گره‌های پایانی دارای دو نوع صفر و یک بوده و تنها یک یال خروجی دارند. در شکل(۴) عملکرد منطقی یک گره تصمیم‌گیری و در شکل(۵) یک تابع منطقی و نمایش OBDD کامل آن نشان داده شده است.



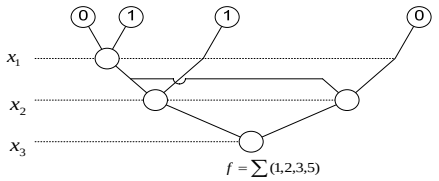
شکل(۴): عملکرد بولی یک گره تصمیم گیری.

با توجه به اینکه تابعی با n ورودی دارای 2^n ترکیب مختلف ورودی می‌باشد، نمایش OBDD کامل آن به 2^n-1 عدد گره تصمیم‌گیری نیاز دارد. نمایش BDD یک تابع منطقی می‌تواند توسط قانون‌های حذف گره و حذف زیر درخت ادغام[4] ، ساده شده و تعدادی از گره‌های آن حذف گردد.



شکل(۵): نمایش OBDD کامل برای یک تابع منطقی.

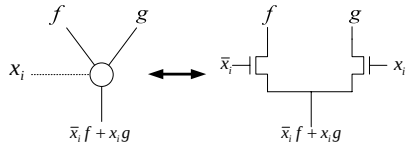
با اعمال دو قانون فوق بر روی یک OBDD کامل،[۷] ROBDD بدست می‌آید بطوریکه هیچ یک از دو قانون فوق دیگر برروی آن قابل اعمال نیست. طبق [5] نمایش ساده شده OBDD یک تابع منطقی منحصر به فرد می‌باشد. در شکل(۶) OBDD ساده شده تابع شکل(۵) نشان داده شده است.



شکل(۶): OBDD ساده شده تابع منطقی f.

با دوبار اعمال قانون حذف گره و یک بار اعمال قانون حذف زیر درخت، فرم کامل OBDD شکل(۵) به فرم ساده شده شکل(۶) تبدیل می‌شود.

با داشتن نمایش ساده شده OBDD یک تابع منطقی می‌توان شبکه‌ی پایین بر DCVS متناظر با آن‌را طراحی نمود. در شکل(۷) نمایش مداری یک گره تصمیم گیری آمده است. در شکل(۸) مدار نهایی DCVS تابع منطقی مفروض نمایش داده شده است.

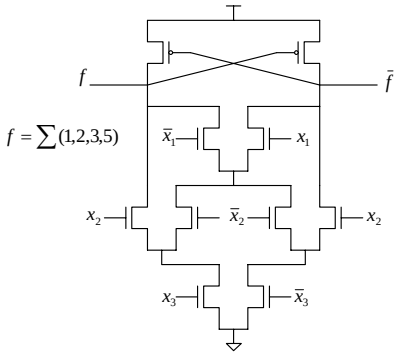


شکل(۷): معادل مداری یک گره تصمیم گیری.

روش OBDD دارای الگوریتم کاملا مشخص و پایان پذیر می‌باشد و از این جهت نسبت به روش‌های قبلی دارای ارجحیت می‌باشد.

۲-۵- طراحی شبکه‌ی پایین بر با استفاده از روش 123dd

در این روش با مدل کردن تابع منطقی به ساختاری بنام 123dd[۸] و سپس انجام مراحل ساده سازی با قوانین مربوطه، شبکه‌ی پایین بر ساده شده بدست می‌آید. جزییات روش فوق در [6] و [7] مطرح گردیده است.

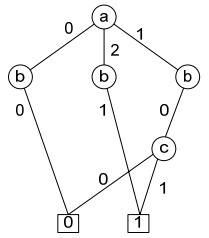


شکل(۸): مدار DCVS ساده شده تابع منطقی f.

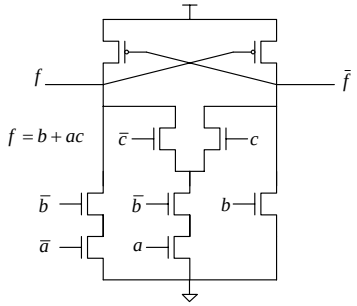
هر 123dd دارای تعدادی گره غیر پایانی است که با دایره نمایش داده می شود. از هر گره غیر پایانی سه نوع یال 0 و 1 و 2 خارج می شود. به هر گره غیر پایانی یک متغییر ورودی تابع منطقی نظیر می گردد که در وسط دایره نشان داده می شود. ورودی هر گره از طریق یکی از انواع سه یال 0 و 1 و 2 به خروجی گره دیگری متصل است، به غیر از گره ریشه که ورودی آن به زمین متصل است. هر 123dd دارای دو گره پایانی صفر و یک است که با مربع نشان داده می شوند و این گره ها دارای خروجی نیستند. گره پایانی 0 (1) در 123dd، متناظر با خروجی f (نقیض f) در شبکه ی پایین بر DCVS هستند. در گره غیر پایانی با متغییر x_i، یال نوع 1 (0) نشان دهنده یک ترانزیستور از نوع n است که به دریچه آن متغییر x_i (نقیض x_i) اعمال شده است. یال از نوع دو به معنی اتصال کوتاه بوده و برای آن ترانزیستوری مورد نیاز نیست. هر گره دارای یک رشته بیتی شامل تعدادی صفر و یک می‌باشد. رشته بیتی گره ریشه همان تابع منطقی نمایش داده شده توسط 123dd می باشد. طول رشته بیتی گره های پایانی برابر یک می‌باشد. تولید ساختار 123dd یک تابع منطقی توسط سه قانون تولید

گره، جابجایی و ادغام انجام می‌گیرد که جزییات آن در[6] بـه تفصـیل آمده است:

در شـکل(۹)، نمـایش 123dd تـابع f=b+ac نشـان داده شـده و درشکل(۱۰) مدار DCVS متناظر با شکل(۹)، به تصویر کشـیده شـده است.



شکل(۹): نمایش 123dd تابع منطقی f=b+ac



شکل(۱۰): مدار DCVS نهایی تابع f با استفاده از روش 123dd.

همانطور که در شکل(۱۰) نیز مشخص است، پیاده سـازی تـابع نشـان داده شده در شکل(۹) نیاز به هفت ترانزیستور دارد کـه در مقایسـه بـا روش OBDD یک ترانزیستور کمتر مصرف می‌نماید.

بر خلاف روش OBDD که برای پیاده سازی گره‌هـای تصـمیم‌گیـری همیشه به دو ترانزیستور نیاز دارد، در روش 123dd بسـته بـه شـرایط ممکن در یک گره دو یا یک یا صـفر ترانزیسـتور مـورد اسـتفاده قـرار گیرد. بطور کلی می‌توان گفـت کـه ایـن روش بـا تغییـر سـاختار گـره تصمیم‌گیری نسبت به روش OBDD می‌توانـد شـبکه‌هـای پـایین بـر بهتری نسبت به روش OBDD تولید نماید.

۲-۶- روش پیشنهادی

روش پیشنهادی که ما ارائه خواهیم کرد، مبتنی بر مفهوم *توابع هم ارز* بوده و مشخصه‌ی اصلی آن دارا بودن الگوریتم کاملاً مشـخص و پایـان پذیر جهت یافتن درخت پایین بر بهینه می‌باشد. لازم به توضیح اسـت که پیاده سازی روش فوق نیز در[۸] انجام گردیـده اسـت و توسـط آن می‌توان عملکرد روش را مورد بررسی قرار داد.

۲-۷- توابع هم ارز

توابع منطقی $f(x_{n-1},x_{n-2},\ldots,x_0)$ و $g(y_{n-1},y_{n-2},\ldots,y_0)$ را هـم ارز گوینـد اگر و تنها اگر ماتریس $T_{n\times n}$ با مشخصات زیر وجود داشته باشد:

- در هر سطر ماتریس T تنها یک درایه برابر یک و بقیه درایه ها برابر صفر باشد.
- در هر ستون ماتریس T تنها یک درایه برابر یک بوده و بقیه درایه ها برابر صفر باشد.
- به نحوی که روابط زیر برقرار باشد :

$$\begin{bmatrix} x_{n-1} & x_{n-2} & \cdots & x_0 \end{bmatrix} \begin{bmatrix} t_{1,1} & t_{1,2} & \cdots & t_{1,n} \\ t_{2,1} & t_{2,2} & \cdots & t_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ t_{n,1} & t_{n,2} & \cdots & t_{n,n} \end{bmatrix} = \begin{bmatrix} z_{n-1} & z_{n-2} & \cdots & z_0 \end{bmatrix} \quad (۱)$$

$$f(x_{n-1},x_{n-2},\ldots,x_0)=g(z_{n-1},z_{n-2},\ldots,z_0) \quad (۲)$$

با توجه به وارون پذیر بودن ماتریس T می توان ثابت نمود که اگر تابع f هم ارز تابع g باشد آنگاه تابع g نیز هم ارز تابع f است.

ماتریس T که یک ماتریس مربع می‌باشـد، مـاتریس تبـدیل هـم ارزی نامگذاری شده است. هـر تـابع منطقـی f بـا خـودش هـم ارز اسـت و ماتریس تبدیل آن یک ماتریس واحد می باشـد. بـا توجـه بـه خـواص ضرب ماتریس‌ها ثابت می شود که در صورتیکه تـابع f هـم ارز تـابع g باشد و تابع h هم ارز تابع f باشد آنگاه تابع g هم ارز تـابع h اسـت. بـا توجه به توضیحات فوق می توان گفت که تمامی توابع هم ارز یک تابع منطقی در یک مجموعه قرار دارند که این مجموعه، *مجموعه توابع هم-ارز* نامگذاری شده است. با داشتن هر تـابع منطقـی از مجموعـه توابـع هم ارز یک تابع منطقی می توان با انتخاب ماتریس تبدیل مناسـب بـه هر یک از توابع دیگر آن مجموعه دست یافت.

مثال: سه تابع منطقی f_1، f_2 و f_3 بصورت زیر مفروض هستند :

$$f_1(x_2,x_1,x_0)=\sum(2,3,5,6,7) \quad (۳)$$

$$f_2(y_2,y_1,y_0)=\sum(3,4,5,6,7) \quad (۴)$$

$$f_3(z_2,z_1,z_0)=\sum(1,3,5,6,7) \quad (۵)$$

تابع f_1 هم ارز تابع f_2 می باشد زیرا با ماتریس $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$ داریم:

$$f_1(x_2,x_1,x_0)=f_2(x_2,x_0,x_1) \quad (۶)$$

و تابع f_1 هم ارز تابع f_3 می باشد زیرا با ماتریس $\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ داریم:

$$f_1(x_2,x_1,x_0)=f_3(x_2,x_0,x_1) \quad (۷)$$

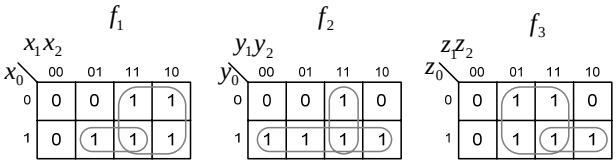
جهت بررسی صحت روابط (۶) و (۷) جدول صـحت توابـع f_1، f_2 و f_3 در جدول(۱) نشان داده شده است. در شکل(۱۱) بـرای بدسـت آوردن عبارت منطقی توابع f_1، f_2 و f_3 از جدول کارنو استفاده شده است.

x_2	x_1	x_0	f_1
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

y_2	y_1	y_0	f_2
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

z_2	z_1	z_0	f_3
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

جدول(۱): جدول صحت سه تابع منطقی f_1، f_2 و f_3.



شکل(۱۱): جدول کارنو توابع f_1، f_2 و f_3.

با استفاده از جدول کارنو برای توابع فوق روابط زیر بدست می آید.

$$f_1=x_1+x_0x_2 \quad (۸)$$

$$f_2=y_0+y_1y_2 \quad (۹)$$

$$f_3=z_2+z_1z_0 \quad (۱۰)$$

همانطور که از روابطـه (۸)،(۹) و (۱۰) مشـخص اسـت، هـر سـه تـابع منطقی f_1، f_2 و f_3 دارای ماهیت یکسانی هستند و در حقیقت هر سـه، عبارت منطقی یکسانی را پیاده سازی می‌نمایند.

۲-۷-۱- تعداد توابع هم ارز یک تابع منطقی

با توجه به ساختار ماتریس تبدیل T در رابطه (۱) مشـخص اسـت کـه برای تابعی با n متغییر ورودی تعداد (n!) عدد ماتریس تبدیل مختلف وجود دارد و با توجه به تعریف ارائه شده برای توابع هم ارز، به ازاء هـر ماتریس تبدیل یک تابع هم ارز وجود دارد. هر یک از این ماتریس‌هـای تبدیل بصورت T_i که در آن $0 \le i \le (n!-1)$ نشان داده مـی‌شـود و هر یک از توابع هم ارز نظیر هر یک از T_i ها را بصورت f_i نشـان مـی-دهیم.

از میان مجموعه ماتریس‌هـای تبـدیل، برخـی منجـر بـه تولیـد توابـع یکسانی می‌گردد. این دسته از ماتریس‌های تبدیل نمی‌توانند در ایجـاد شبکه‌ی پایین بر بهینه موثر باشند و می‌توان از آنها صـرف نظـر نمـود. زیرمجموعـه‌ای از مجموعـه‌ی توابـع هـم ارز یـک تـابع دارای اهمیـت هستند که متمـایز از یکـدیگر بـوده و دارای جـدول صـحت متفـاوتی باشند. این زیرمجموعه، مجموعه *توابع هم ارز متمایز* نام گـذاری شـده است.در مورد تابع f_1 مطرح شده در مثال قبل، ماتریس های تبـدیل $\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}$ و $\begin{bmatrix}0&0&1\\0&1&0\\1&0&0\end{bmatrix}$ منجر به تولید مجدد f_1 می‌گردنـد و مـاتریس $\begin{bmatrix}0&1&0\\0&0&1\\1&0&0\end{bmatrix}$ منجر به تولید مجـدد f_2 و مـاتریس $\begin{bmatrix}0&0&1\\1&0&0\\0&1&0\end{bmatrix}$ منجـر بـه تولید مجـدد f_3 مـی‌گـردد. در مـورد تـابع f_1 بـا سـه متغییـر ورودی مجموعه توابع هم ارز دارای شش عضو است که تنها سـه تـابع هـم ارز متمایز در آن وجود دارد.

دلیل ایجاد توابع هم ارز برابر در مورد مثال فوق وجود جزء متقـارن در ایجاب کننده‌های نخستین (PI ها) تابع f_1 می‌باشد.

تابع $f(x_{n-1},x_{n-2},\ldots,x_0)$ متقارن[۹] است اگر و تنها اگر رابطه (۱۱) به ازاء هر i و j بین صفر و n-1 صدق نماید:

$$f(x_{n-1},x_{n-2},\ldots,\bar{x}_i,\ldots,x_j,\ldots,x_0)=f(x_{n-1},x_{n-2},\ldots,x_i,\ldots,\bar{x}_j,\ldots,x_0) \quad (۱۱)$$

برای توابع متقارن تمامی ماتریس‌های تبدیل، منجر به تولید خود تـابع مـی‌گـردد. تقـارن ممکـن اسـت بـرای زیرمجموعـه‌ای از مجموعـه‌ی متغییرهای ورودی یک تابع منطقی وجود داشته باشد. به عنوان مثـال در تابع f_1،ورودی های 2^0 و 2^2 دارای تقارن می‌باشد و همانطور که در رابطه (۸) مشخص است علت این امر وجود تابع منطقـی متقـارن OR در میان PI های تابع f_1 می‌باشد.

۲-۷-۲- تولید توابع هم ارز متمایز

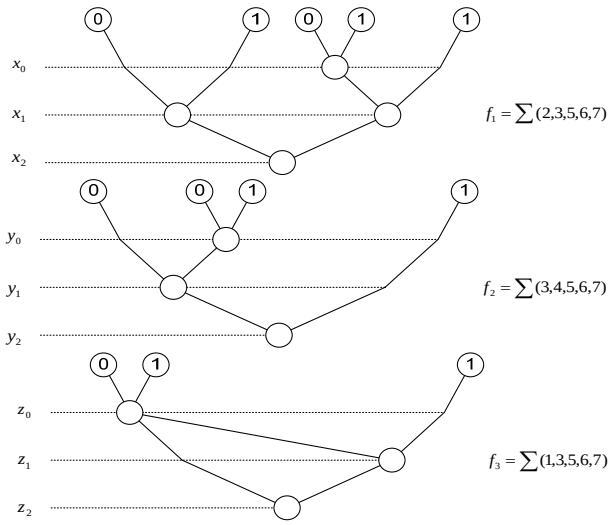
به منظور تولید تمام توابع هم ارز متمایز، ابتدا تمام ماتریس‌های $T_{n \times n}$ را که دارای دو خاصیت ماتریس تبدیل هسـتند، تولیـد مـی‌شـود. هـر ماتریس تبدیل بصورت T_i که در آن $0 \le i \le (n!-1)$ است، نشـان داده می‌شود. با اعمال ماتریس‌هـای تبـدیل بـر روی ورودی‌هـای تـابع منطقی، توابع هم ارز تابع منطقی ورودی بدست می‌آید. توابـع هـم ارز حاصل از اعمال ماتریس تبدیل T_i را با f_i نمایش می‌دهیم. برای مثـال تمام ماتریس‌های تبدیل برای تابع منطقی با سـه متغییـر برابـر شـش است که عبارتند:

$$T_0=\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix},\quad T_1=\begin{bmatrix}1&0&0\\0&0&1\\0&1&0\end{bmatrix},\quad T_2=\begin{bmatrix}0&1&0\\1&0&0\\0&0&1\end{bmatrix},\quad T_3=\begin{bmatrix}0&0&1\\1&0&0\\0&1&0\end{bmatrix},\quad T_4=\begin{bmatrix}0&1&0\\0&0&1\\1&0&0\end{bmatrix},\quad T_5=\begin{bmatrix}0&0&1\\0&1&0\\1&0&0\end{bmatrix} \quad (۱۲)$$

به علت آنکه ماتریس T_0 ماتریس واحد می‌باشد، تابع هم ارز f_0 ، همان تابع ورودی می‌باشد. پس از تولید تمام توابع هم ارز با حـذف f_i هـای تکراری مجموعه‌ی توابع هم ارز متمایز بدست می‌آید.

۲-۸- کاربرد توابع منطقی هم ارز در طراحی شبکه‌ی پایین بر بهینه

همانطور که در بخشهای گذشته گفته شد، با اعمال هر یک از روشهای گفته شده می توان درخت پایین بر ساده شده ای را تولید نمود. اعمال قوانین ساده‌سازی در هر روش بستگی مستقیم به تابع منطقی و بطـور دقیق‌تر به فرم نمایش مجموع حاصلضرب تابع منطقـی دارد. از طرفـی همانطور که در مثال قبـل در روابـط (۳) و (۴) و (۵) مشـخص اسـت، هریک از توابع متعلق بـه مجموعـه توابـع هـم ارز متمـایز، دارای فـرم نمایش *مجموع حاصلضرب* متفـاوتی هسـتند. ایـن تفـاوت در نمـایش مجموع حاصلضرب می تواند منجـر بـه اسـتفاده بیشـتر قـوانین سـاده سازی در الگوریتم‌های ساده‌سـازی گـردد. در شـکل(۱۲) زیـر درخـت پایین بر ساده شده توابع مطرح شده در روابـط (۳)،(۴) و (۵) توسـط روش OBDD آمده است.

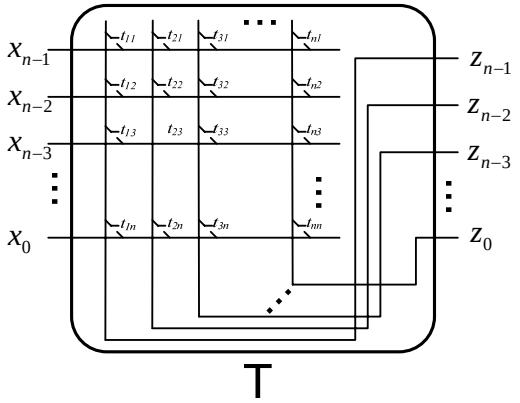


شکل(۱۲): درخت پایین بر ساده شده توسط روش OBDD سه تابع هم ارز.

همانطور که در شکل(۱۲) مشخص است پیاده‌سازی درخت پایین بر ساده شده f_1 به چهار گره تصمیم‌گیری (هشت ترانزیستور) نیاز دارد و برای پیاده سازی درخت پایین بر ساده شده توابع f_2 و f_3 به سه گره تصمیم گیری (شش ترانزیستور) نیاز می‌باشد.

نکته بسیار مهم آن است که توابع هم ارز متمایز از نظر عملکرد با یکدیگر مساوی نبوده و برای آنکه بتوان یکی از آنها را به جای دیگری استفاده نمود، به ماتریس تبدیل مربوطه نیز نیاز می‌باشد.البته با داشتن ماتریس تبدیل تابع به هم ارز آن، می‌توان از هم ارز تابع بجای تابع استفاده نمود. پیاده سازی مداری تابع تبدیل در شکل(۱۳) نشان داده شده است.

در شکل(۱۳) یک بودن مقدار t_{ij} به معنی برقراری اتصال و صفر بودن آن به منزله قطع بودن اتصال در موقعیت مربوطه می باشد که در آن t_{ij} درایه موجود در سطر i ام و ستون j ام ماتریس تبدیل T می‌باشد.



شکل(۱۳): نمایش مداری برای ماتریس تبدیل T.

۲-۹- الگوریتم یافتن شبکه ی پایین بر بهینه در روش پیشنهادی

ورودی الگوریتم، نمایش مجموع حاصلضرب تابع منطقی بوده و خروجی آن ساده ترین شبکه ی پایین بر DCVS می باشد. روش پیشنهادی ابتدا مجموعه توابع هم ارز متمایز را برای تابع منطقی داده شده بدست می آورد و سپس با اعمال روش‌های موجود ساده‌سازی که در مورد آنها بحث شد، ساختار شبکه‌ی پایین بر ساده شده برای تمام توابع هم ارز را بدست می‌آورد و با مقایسه آنها با یکدیگر بهترین شبکه‌ی پایین بر انتخاب می شود. با توجه توضیحات فوق مشخص است که شبکه‌ی پایین بر بدست آمده توسط این الگوریتم دارای تعداد ترانزیستور کمتر و یا مساوی روش‌های قبلی است.

در این روش می‌توان بطور همزمان از چند روش ساده‌سازی استفاده نمود بدین ترتیب که برای هر تابع هم ارز چند روش را اعمال و نتیجه آن ثبت می‌شود و در نهایت از بین تمام پاسخ‌های بدست آمده، بهترین شبکه‌ی پایین بر را انتخاب می‌گردد.

در جدول (۲) تعداد ترانزیستورهای لازم برای توابع:

$$f_1 = \sum(0,2,3,8,10,11,14,15,16,18,19,20,23,24,26,27,31)$$
$$f_2 = \sum(0,1,2,5,6)$$
$$f_3 = \sum(0,1,2,6,8,9,11,12,14,15)$$

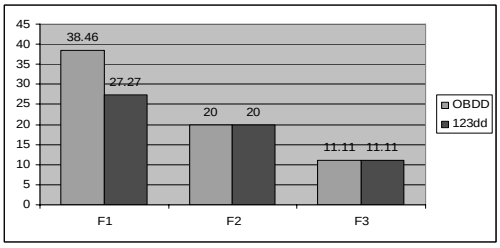
براساس متدهای OBDD و 123dd و روش پیشنهادی نشان داده شده است.

f_3	f_2	f_1	
18	**10**	**26**	OBDD
18	**10**	**22**	123dd
16	**8**	**16**	روش پیشنهادی

جدول(۲) مقایسه ی روشهای OBDDو123ddو روش پیشنهادی

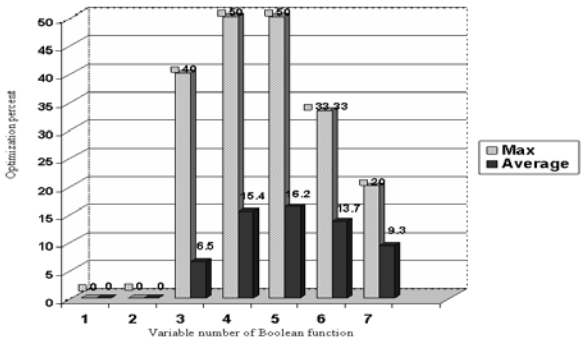
توجه داشته باشید که توابع مذکور از آن جهت مورد اهمیت هستند که در [6] و [4] برای ارزیابیِ کاراییِ روش‌های ارائه شده از آنها استفاده شده است.

در نمودار رسم شده‌ی شکل(۱۴) نیز درصد بهینه سازی روش پیشنهادی را نسبت به روشهای OBDD و 123dd برای توابع مذکور ملاحظه می فرمایید.



شکل(۱۴) نیز درصد بهینه سازی روش پیشنهادی نسبت به روشهای OBDD و 123dd

در شکل (۱۵) نیز درصد بیشترین بهینه‌سازی و بهینه‌سازی متوسط را برای توابع بولی ۱ تا ۷ متغیره ملاحظه می‌نمایید لازم به ذکر است که برای توابع ۴ متغیره و بیشتر درهر آیتم تعداد ده‌هزار تابع بولی تست شده است.



شکل(۱۵) درصد بهینه سازی برای توابع ۱ تا ۷ متغیره

نتیجه گیری

با استفاده از مفهوم توابع هم ارزش می توان بـا اسـتفاده از روش هـای موجود برای طراحی شبکه‌ی پـایین بـر بهینـه مـدارهای DCVS، بـه مدارهایی با تعداد ترانزیستور کمتـر، دسـت یافـت. بـا اسـتفاده از ایـن تکنیک می‌توان حالت‌های بیشتری را بـرای اسـتفاده از قـوانین سـاده-سازی ایجاد نمود و با توجه به وجود الگوریتم مشـخص بـرای آن مـی-تـوان از آن در بسـته‌هـای نـرم افـزاری طراحـی کـامپیوتری (CAD) استفاده نمود. این روش در الگوریتم کاری خود از یـک یـا چنـد روش ساده سازی مختلف بطور همزمان استفاده می‌نماید. برای مثال می‌توان روش 123dd را همزمان با OBDD ویا حتی سایر روش‌ها[9] استفاده نمود و بهترین شبکه در بین تمام توابـع هـم ارز را در نهایـت انتخـاب نمود. با توجه به الگوریتم مورد اسـتفاده مشـخص اسـت کـه شـبکه‌ی پایین بر تولید شده توسط این روش همیشه برابر و یا بهتر از هر یک از روش‌های ساده سازی مورد استفاده است.

۲-۱۰- گام بعدی

در برخی شرایط تابع منطقی، یک تابع ساده تک خروجـی نبـوده و بـه ازاء هر بردار ورودی یک بردار خروجی تولید می شود (فرم کلـی تـابع منطقی بصورت $\{0,1\}^n \xrightarrow{f} \{0,1\}^m$ است). به عنوان مثـال یـک ROM بـا دریافـت یـک بـردار ورودی (آدرس) یـک بـردار خروجـی (محتوای خانه حافظه) را بر می‌گرداند یا یک کمپرسـور 2-4 کـه دارای پنج ورودی و سه خروجی می‌باشد. مدارهایی که از این دسـته هسـتند معمولاً توسط روش‌های جـزء بنـدی یعنـی تقسـیم بـه قسـمت هـای کوچکتری که روش‌های ابتکاری قادر به تحلیل آن باشند، طراحی می-شوند در صورتیکه ممکن است با یکپارچه دیدن مدار بتوان قسمت‌های مشترک بیشتری را در هم ادغام و به مدارها به مراتب ساده‌تری برسیم.

با روش‌های موجود می‌توان برای هر یـک از m خروجـی تـابع f،یـک تابع f_i ($1 \le i \le m$) بصورت $\{0,1\}^n \xrightarrow{f_i} \{0,1\}$ ایجاد نمـاییم بطوریکه مجموعه توابع f_i بتواند تابع f را پیـاده سـازی نمایـد و البتـه برای پیاده سازی توابع f_i می‌توان از الگوریتم مطرح شده جهت سـاده-سازی هرچه بیشتر استفاده نمود. اشکال این روش آن است که توابع f_i ماهیت مستقلی نسبت به هم دارنـد و بـه همـین دلیـل قـوانین سـاده سازی موجود نمی‌توانند عمل بهینه سازی را برای قسمت‌های مشترک بین توابع f_i انجام دهند.

قدم بعدی تکمیل الگوریتم مطرح شده به نحوی است که بتوانـد عمـل ساده‌سازی را بین دسـته‌ای از توابـع کـه دارای ورودی-هـای یکسـانی هستند انجام دهد. به نظر می‌رسد کـه الگـوریتم مطـرح شـده در ایـن نوشتار زمانی که ساده‌سازی دسته‌ای از توابـع مطـرح باشـد، بـه علـت ضرب شدن تعداد حالت‌های توابـع هـم ارز در یکـدیگر، از نظـر مرتبـه زمانی، غیر عملی باشد.

مراجع

[1] K.M. Chu and D.L. Pulfrey, 1987, "A Comparison of CMOS Circuit Techniques: Differential Cascode Voltage Switch Logic Circuits Versus Conventional Logic", IEEE Journal of Solid-State Circuits, vol. SC-22, no. 4, pp. 528-532

[2] L. Heller, W. Griffin, J. Davis and N. Thoma, 1984, "Cascode voltage switch logic: a differential CMOS logic family". Proceedings of IEEE International Solid-State Circuit, Conference, pp. 16-17.

[3] K.M. Chu and D.L. Pulfrey, 1986, "Design Procedures for Differential Cascode Voltage Switch Logic Circuits", IEEE Journal of Solid-State Circuits, vol. SC-21, no. 4, pp.1082-1087

[4] T. Karoubalis, G.Ph. Alexiou, N. Kanopoulos,1995, "Optimal synthesis of differential cascode voltage switch (DCVS) logic circuits using ordered binary decision diagrams (OBDDs)", Design Automation Conference, pp. 282-287

[5] S. Bandyopadhyay, A. Jaekel ,G.A Jullien, "A method for synthesizing area efficient multilevel PTL circuits", 1997, Tenth International Conference on VLSI Design, pp. 516-518

[6] A. Jaekel, 1997, "A New Model for Constructing DCVS Trees", IEEE 1997 Canadian Conference, vol. 2, pp. 641-645

[7] A. Jaekel, S. Bandyopadhyay, G.A. Jullien, 1998,"Design of dynamic pass-transistor logic circuits using 123 decision diagrams", IEEE Transactions on Circuits and Systems , vol. 45, no. 11, pp. 1172-1181.

[۸] محتشمی، حسین، **" بررسی،تحلیل و طراحی مدارات DCVS بهینه"**، پایان نامه کارشناسی ارشد ، دانشگاه شهید بهشتی، شهریور ۱۳۸۴.

[9] O.Kavehie, K.Navi, T.Nikoubin, M.Rouholamini"A Novel DCVS Tree Reduction Algorithm",Integrated Circuit Design and Technology,ICICDT'06. 2006 IEEE International Conference, pp.1-7.

زیرنویس‌ها

[1] Differential Cascode Current Switch Logic
[2] Quine-McCluskey
[3] Primary Implicant
[4] Ordered Binary Decision Diagram
[5] Binary Decision Diagram
[6] Decision Variable
[7] Reduced Ordered Binray Decision Diagram
[8] 123 Decision Diagram
[9] Symmetric functions