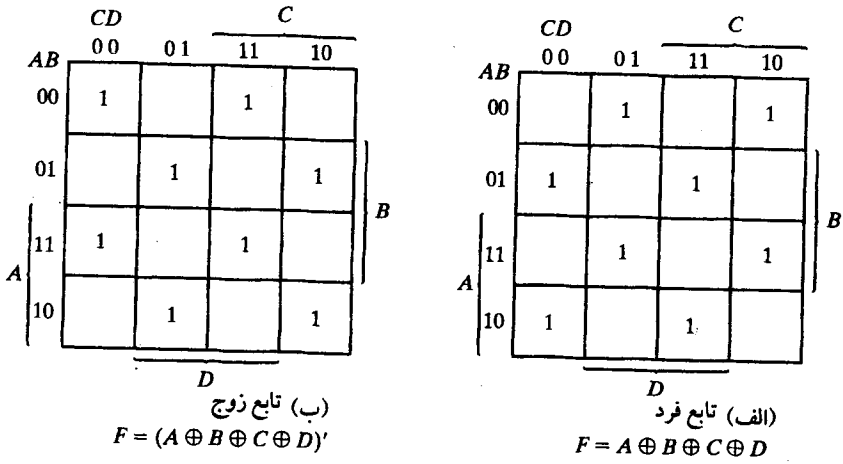


شکل ۳۵-۳. نقشه برای تابع XOR چهار متغیره

این تابع را بدست آوریم.

$$\begin{aligned} A \oplus B \oplus C \oplus D &= (AB' + A'B) \oplus (CD' + C'D) \\ &= (AB' + A'B)(CD + C'D') + (AB + A'B')(CD' + C'D) \\ &= \Sigma(1, 2, 4, 7, 8, 11, 13, 14) \end{aligned}$$

برای تابع بول چهار متغیره 16 مینترم وجود دارد. نیمی از مینترم‌ها دارای تعداد فردی 1 در مقادیر عددی خود هستند؛ نیمه دیگر دارای تعداد زوجی 1 در مینترم می‌باشند. هنگام ترسیم تابع در نقشه، مقدار عدد دودویی هر مینترم از اعداد سطر و ستون مربعی که مینترم را نمایش می‌دهد بدست می‌آید. نقشه شکل ۳۵-۳ (الف) مربوط به تابع XOR چهار متغیره است. این یک تابع فرد است زیرا مقادیر دودویی همه مینترم‌ها تعداد فردی 1 دارند. متمم یک تابع فرد هم یک تابع زوج است. طبق شکل ۳۵-۳ (ب) تابع زوج چهار متغیره هنگامی 1 است که تعداد زوجی از متغیرها در مینترم برابر 1 باشد.

تولید و چک توازن

توابع XOR در سیستم‌هایی که به کدهای عیب‌یاب و تصحیح‌کننده خطا نیاز دارند بسیار مفیدند. همانطور که در بخش ۷-۱ ملاحظه شد، یک بیت توازن به منظور تشخیص خطا در حین انتقال اطلاعات دودویی به آن اضافه می‌شود. بیت توازن، بیتی اضافی است که با پیام دودویی همراه می‌شود تا تعداد 1ها را زوج یا فرد کند. پیام، از جمله بیت توازن، ارسال و سپس در مقصد برای تشخیص خطا چک می‌شود. اگر توازن چک شده با آنچه ارسال شده است تطابق نداشت. یک خطا اعلام می‌گردد. مداری که بیت توازن را در فرستنده تولید می‌کند، مولد توازن نامیده می‌شود. مداری که توازن را در سمت گیرنده چک می‌کند چک کننده توازن خوانده می‌شود.

جدول ۴-۳. جدول درستی مولد توازن زوج

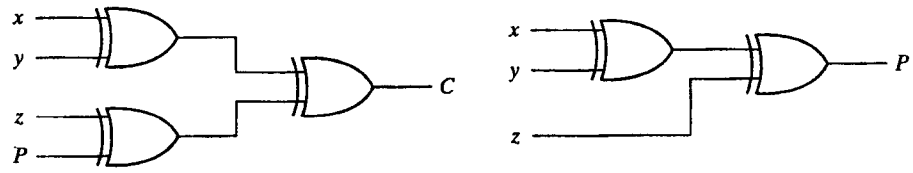
پیام سه بیتی			بیت توازن
x	**y**	**z**	**P**
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

به عنوان مثال، فرض کنید بخواهیم یک پیام سه بیتی را همراه با یک بیت توازن زوج ارسال کنیم. جدول ۴-۳، جدول درستی را برای مولد توازن نشان می‌دهد. سه بیت x و y و z که پیام را تشکیل می‌دهند ورودی به مدار هستند. بیت توازن P، خروجی است. برای توازن زوج، بیت P باید طوری باشد که تعداد کل 1 ها را زوج کند (از جمله P). از جدول درستی می‌بینیم که P یک تابع فرد را تشکیل می‌دهد زیرا برای مینترم‌هایی که تعداد فردی 1 دارند باید برابر 1 شود. بنابراین P به صورت یک تابع XOR سه متغیره بیان می‌شود.

$$P = x \oplus y \oplus z$$

نمودار منطقی مولد توازن در شکل ۳۶-۳ (الف) ملاحظه می‌شود.

سه بیت پیام، همراه با بیت توازن به مقصد ارسال می‌شوند و در آنجا به مدار چک کننده توازن برای چک کردن خطای محتمل به هنگام ارسال، وارد می‌گردند. چون اطلاعات با توازن زوج ارسال شده است، چهار بیت دریافتی باید تعداد زوجی 1 داشته باشد. خطا در حین انتقال هنگامی رخ می‌دهد که چهار بیت دریافتی تعداد فردی 1 دارد، و این به معنی رخداد خطا در حین انتقال است. خروجی چک کننده توازن که با C مشخص شده است به هنگام رخداد خطا برابر 1 می‌شود. یعنی اگر چهار بیت دریافتی تعداد فردی 1 داشته باشد خطا رخ داده است. جدول ۵-۳ جدول درستی برای چک کننده توازن زوج است. با توجه به آن ملاحظه می‌شود که تابع C متشکل از هشت مینترم با مقادیر دودویی دارای تعداد فردی 1 است. این مطالب مربوط به نقشه ۳۵-۳ (الف) است که تابع فرد را نشان می‌دهد. می‌توان



(الف) مولد توازن زوج 3 بیتی (ب) چک کننده توازن زوج 4 بیتی

شکل ۳۶-۳. نمودار منطقی مولد و چک کننده توازن

جدول ۵-۳. جدول درستی چک کننده توازن زوج

چهار بیت دریافتی				چک خطای توازن
x	y	z	P	C
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

چک کننده توازن را با گیت‌های XOR پیاده‌سازی کرد:

$$C = x \oplus y \oplus z \oplus P$$

نمودار منطقی چک کننده توازن در شکل ۳۶-۳ (ب) ملاحظه می‌گردد.

لازم به تذکر است که مولد توازن را با مدار شکل ۳۶-۳ (ب) نیز می‌توان تولید کرد به شرطی که ورودی P به منطق 0 متصل گردد و خروجی نیز با P نام‌گذاری شود. دلیل این است که $z \oplus 0 = z$ بوده و موجب می‌شود تا z از گیت بدون تغییر عبور کند. مزیت این است که برای هر دو مدار تولید و چک کننده توازن از یک مدار مشابه می‌توان استفاده کرد.

با توجه به مثال قبل واضح است که تابع مولد توازن و نیز چک کننده دارای نیمی از کل مینترم‌ها هستند و مقادیر عددی آنها تعداد زوج یا فردی 1 دارند. در نتیجه می‌توان آنها را با گیت‌های XOR پیاده‌سازی کرد. تابعی با تعداد زوجی 1 متمم تابع فرد است. این تابع با XOR پیاده‌سازی می‌شود ولی گیت واقع در خروجی باید XNOR باشد تا متمم لازم را تولید نماید.

۳-۹ زبان توصیف سخت‌افزاری (HDL)

زبان توصیف سخت‌افزاری، زبانی است که سخت‌افزار سیستم‌های دیجیتال را به فرم متنی توصیف می‌نماید. در واقع این زبان، یک زبان برنامه‌نویسی است، ولی خصوصاً حول توصیف ساختارهای سخت‌افزاری و رفتار آنها بنا نهاده شده است. می‌توان از آن برای نمایش نمودارهای منطقی، عبارات بولی و دیگر مدارهای دیجیتال پیچیده استفاده کرد. HDL به عنوان یک زبان مستندسازی برای نمایش و

مستند کردن سیستم‌های دیجیتال به کار می‌رود به نحوی که قابل خواندن به وسیله انسان‌ها و کامپیوترها می‌باشد. این زبان به عنوان زبان تبادل بین دو طراح هم به کار می‌رود. محتوای زبان به طور مؤثر و نیز به سادگی قابل ذخیره، بازیابی و پردازش به وسیله نرم‌افزار کامپیوتر است. در پردازش HDL دو کاربرد وجود دارد: شبیه‌سازی و سنتز.

شبیه‌سازی منطقی نمایشی از ساختار و رفتار یک سیستم منطقی دیجیتال به کمک کامپیوتر است. یک شبیه‌ساز توصیف HDL را تفسیر کرده و یک خروجی قابل خواندن مانند نمودار زمانی، تولید می‌نماید و بدین وسیله رفتار سخت افزار را قبل از ساخت پیش بینی می‌نماید. HDL امکان تشخیص خطای عملیاتی در طراحی را بدون نیاز به خلق فیزیکی آن، فراهم می‌سازد. خطاهایی که در حین شبیه‌سازی شناسایی می‌شوند را می‌توان با اصلاح عبارت مربوطه در HDL رفع کرد. امکاناتی که عملیات طرح را تست می‌کند، **برنامه تست** می‌نامند. بنابراین برای شبیه‌سازی یک سیستم، طرح ابتدا در HDL توصیف شده و سپس صحت عمل آن با شبیه‌سازی طرح و تست آن به وسیله برنامه تست که در HDL نوشته می‌شود، تحقیق می‌گردد.

سنتز منطقی فرآیندی است که طی آن از قطعات و اتصال بین آنها به نام netlist در مدل سیستم دیجیتالی که در HDL توصیف شده است لیستی تهیه می‌گردد. netlist سطح گیت را می‌توان در ساخت یک مدار مجتمع یا طرح بورد مدار چاپی به کار برد. سنتز منطقی مشابه با کامپایل یک برنامه در یک زبان سطح بالاست. تفاوت در این است که، در عوض تولید کد منتج، یک بانک اطلاعاتی تولید می‌نماید که در آن دستورالعمل‌های ساخت یک قطعه سخت‌افزار دیجیتال فیزیکی توصیف شده با کد HDL آمده است. سنتز منطقی بر روال‌هایی مبتنی است که مدارهای دیجیتال را پیاده‌سازی می‌کنند و شامل آن بخش از یک طراحی دیجیتال است که قابل اتوماتیک شدن با نرم‌افزار کامپیوتر باشد.

در صنعت HDL های انحصاری متعددی وجود دارند که به وسیله کمپانی‌ها برای طراحی یا کمک به طراحی مدارهای مجتمع ساخته شده‌اند. دو استاندارد HDL به وسیله IEEE پشتیبانی می‌شوند: VHDL و Verilog HDL . VHDL یک زبان تحت کنترل وزارت دفاع بود که در حال حاضر به صورت تجاری و در دانشگاه‌ها استفاده می‌شود. Verilog به عنوان یک زبان انحصاری که به وسیله کمپانی Cadence Data System ارتقاء یافت، ولی بعد کنترل آن را به مجموعه‌ای از کمپانی‌ها به نام Open Verilog International (OVI) محول کرد. VHDL نسبت به Verilog زبان سخت‌تری است. چون Verilog برای یادگیری ساده‌تر است، ما آن را در این کتاب انتخاب کرده‌ایم. با این وجود، توصیف‌های Verilog HDL لیست شده در سرتاسر این کتاب تنها درباره Verilog نیست، بلکه معرفی مفهوم نمایش سیستم‌های دیجیتال به کمک کامپیوتر به وسیله نوعی زبان توصیف سخت‌افزاری است.

نمایش مدول

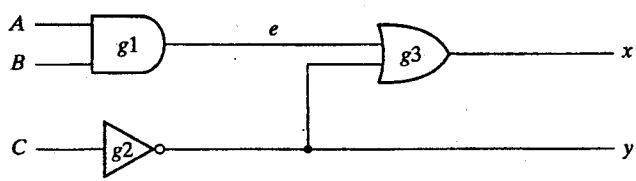
Verilog HDL دارای دستور زبانی است که دقیقاً ساختارهای مُجازی که در زبان می‌توانند به کار روند را توصیف می‌نماید. خصوصاً، Verilog حدود 100 کلمه کلیدی از پیش‌تعریف شده، حروف

کوچک و شناسه‌هایی دارد که ساختار زبان را تعریف می‌کنند. مثال‌هایی از کلمات کلیدی عبارتند از module، endmodule، input، output، wire، and، or، not و غیره. هر متن بین دو اسلش (//) و انتهای خط به عنوان یک توضیح تفسیر می‌گردد. فاصله‌های Blank و نام‌ها حساس به اندازه هستند، و این بدان معنی است که حروف بزرگ و کوچک با هم متفاوتند. در Verilog مدول یک بلوک ساختاری است. این دستور با کلمه کلیدی module آغاز و با کلمه کلیدی endmodule پایان می‌یابد. اکنون برای تشریح بعضی از مفاهیم زبان مثال ساده‌ای را تشریح می‌کنیم.

توصیف HDL مدار شکل ۳۷-۳ در مثال ۱-۳ HDL نشان داده شده است. خطی که دو اسلش دارد توضیحات است و عمل مدار را توضیح می‌دهد. دومین خط، مدول را همراه با نام و لیستی از پورت‌ها مشخص می‌کند. نام (در اینجا smpl–circuit) یک شناسه است که برای ارجاع به مدول به کار رفته است. شناسه‌ها نام‌هایی هستند که به متغیرها داده می‌شوند و به این ترتیب در طراحی قابل ارجاع می‌گردند. آنها از کاراکترهای الفبا عددی و زیرخط (_) ساخته می‌شوند و حساس به اندازه‌اند. شناسه‌ها باید با کاراکتر الفبایی و یا خط تیره شروع شوند. آنها را نمی‌توان با عدد شروع کرد. Port List رابطی بین مدول برای تبادل اطلاعات (مخابره) با محیط است. در این مثال پورت‌ها، ورودی‌ها و خروجی‌های مدارند. Port List بین پرانتزها محصور شده و از ویرگول برای جدا کردن عناصر لیست استفاده می‌شود. عبارت با نقطه ویرگول (;) پایان می‌یابد. همه کلمات کلیدی که باید به حروف کوچک باشند به منظور وضوح با خط پررنگ چاپ می‌شوند، ولی این نیاز زبان نیست. سپس input و output بیان می‌دارند که کدام پورت‌ها ورودی و کدام خروجی هستند. اتصالات درونی در نقش سیم‌ها می‌باشند. مدار دارای یک اتصال داخلی در e بوده و با کلمه کلیدی wire بیان می‌شود. ساختار مدار با گیت‌های اصلی از پیش تعریف شده به عنوان کلمه کلیدی مشخص می‌گردد. معرفی هر گیت با یک نام اختیاری مثل g1، g2 و غیره و به دنبال آن خروجی و ورودی‌هایی که با ویرگول از هم جدا شده و در داخل

مثال ۱-۳، HDL

```
//Description of simple circuit Fig. 3-37
module smpl_circuit(A,B,C,x,y);
   input A,B,C;
   outputx,y;
   wire e;
   and g1(e,A,B);
   not g2(y, C);
   or g3(x,e,y);
endmodule
```



شکل ۳۷-۳. مداری برای نمایش HDL

پرانتزاند، صورت می‌گیرد. همواره خروجی در ابتدا معرفی می‌شود و پس از آن ورودی ذکر می‌گردد. مثلاً گیت OR که g3 نامیده شده، دارای خروجی x و ورودی‌های e و y است. توصیف مدول با کلمه کلیدی endmodule خاتمه می‌یابد. توجه کنید که هر عبارت با یک نقطه ویرگول (;) پایان می‌پذیرد، ولی پس از endmodule نقطه ویرگول گذاشته نمی‌شود.

تأخیر در گیت‌ها

هنگام استفاده از HDL در شبیه‌سازی، گاهی لازم است مقداری تأخیر بین ورودی تا خروجی گیت در نظر گرفته شود. در Verilog تأخیر برحسب واحدهای زمانی و سمبل # معین می‌گردد. ارتباط یک واحد زمانی با زمان فیزیکی با استفاده از رهنمون کامپایلر timescale انجام می‌شود. رهنمون‌های کامپایلر با سمبل " ` " (backqoute) شروع می‌شوند. چنین رهنمونی قبل از اعلان مدول مشخص می‌گردد. مثالی از رهنمون timescale در زیر آمده است.

`timescale 1ns/100ps

عدد اول نشان‌دهنده واحد اندازه‌گیری برای زمان‌های تأخیر است. عدد دوم دقتی که تحت آن تأخیرها گرد شده‌اند را نشان می‌دهد که در این حالت 0.1ns است. اگر timescale مشخص نشود، شبیه‌ساز واحد زمان معینی را، مثل 1ns، پیش‌فرض می‌کند. در این کتاب، واحد زمان پیش‌فرض را انتخاب خواهیم کرد.

مثال ۲-۳ HDL توصیف مثال قبلی را همراه با تأخیر در هر گیت مشخص می‌نماید. گیت‌های AND، OR و NOT به ترتیب زمان تأخیر 30ns، 20ns و 10ns را دارند. اگر مدار شبیه‌سازی شود و ورودی‌ها از 000 به 111 تغییر یابند، خروجی‌ها طبق جدول ۶-۳ تغییر می‌کنند. خروجی وارونگر در y پس از تأخیر 10ns از 1 به 0 تغییر می‌یابد. خروجی گیت AND در e پس از 30ns تأخیر از 0 به 1 تغییر می‌کند. خروجی گیت OR در x در t=30ns از 1 به 0 می‌رود و سپس در t=50ns به 1 باز می‌گردد. در هر دو حالت، تغییر در خروجی گیت OR از تغییری که در 20ns قبل در ورودی‌اش اتفاق می‌افتد، ناشی می‌شود. واضح است که هر چند خروجی x پس از تغییرات ورودی نهایتاً در 1 ثبات پیدا می‌کند، تأخیرهای گیتی قبل از آن برای مدت 20ns یک جرقه منفی ایجاد می‌نمایند.

برای شبیه‌سازی یک مدار با HDL، لازم است ورودی‌ها را برای شبیه‌ساز به مدار اعمال کنیم تا پاسخ خروجی تولید گردد. یک توصیف HDL که محرک را برای یک طرح فراهم می‌کند **برنامه تست**

مثال ۲-۳، HDL

```
//Description of circuit with delay
module circuit_with_delay (A,B,C,x,y);
   input A,B,C;
   output x,y;
   wire e;
   and #(30) g1(e,A,B);
   or #(20) g3(x,e,y);
   not #(10) g2(y,C);
endmodule
```

جدول ۶-۳. خروجی گیت‌ها پس از تأخیر

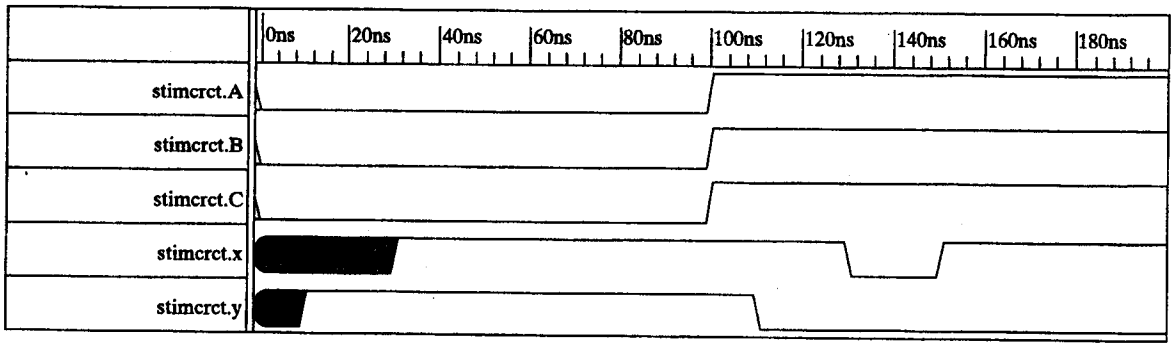
	واحد زمانی	ورودی	خروجی
	(ns)	ABC	y e x
تغییر اولیه	–	000	1 0 1
	–	111	1 0 1
	10	111	0 0 1
	20	111	0 0 1
	30	111	0 1 0
	40	111	0 1 0
	50	111	0 1 1

خوانده می‌شود. نوشتن برنامه‌های تست در انتهای بخش ۱۱-۴ توضیح داده شده است. در اینجا بدون آن که توضیحاتی اضافی را ارائه کنیم روال را با مثال ساده‌ای شرح می‌دهیم. مثال ۳-۳ HDL یک برنامه تست را برای شبیه‌سازی مدار تأخیردار نشان می‌دهد. دو مدول در این برنامه تست لحاظ شده است: مدول محرک و مدول توصیف مدار. مدول محرک stimcrct پورت ندارد. ورودی‌ها به مدار با کلمه کلیدی reg و خروجی نیز با کلمه کلیدی wire معرفی می‌شوند. circuit_with_delay با cwd نام‌گذاری یا ذکر شده است. (واکنش متقابل بین مدول محرک و مدول مدار در شکل ۳۳-۴ نشان داده شده است.)

مثال ۳-۳، HDL

```
//Stimulus for simple circuit
module stimcrct;
reg A,B,C;
wire x,y;
circuit_with_delay cwd(A,B,C,x,y);
initial
   begin
        A = 1'b0; B = 1'b0; C = 1'b0;
     #100
        A = 1'b1; B = 1'b1; C = 1'b1;
     #100  $finish;
   end
endmodule

//Description of circuit with delay
module circuit_with_delay (A,B,C,x,y);
   input A,B,C;
   output x,y;
   wire e;
   and #(30) g1(e,A,B);
   or  #(20) g3(x,e,y);
   not #(10) g2(y,C);
endmodule
```



شکل ۳۸-۳. خروجی شبیه‌سازی شده مثال ۲-۳ HDL

عبارت initial ورودی‌های بین کلمات کلیدی begin و end را مشخص می‌نماید. در آغاز ABC=000 است. (هر یک از A، B و C با 1'b0 تنظیم شده‌اند، و به معنی یک رقم دودویی با مقدار 0 است.) پس از 100ns ورودی‌ها به ABC = 111 تبدیل می‌شوند. پس از 100ns ثانیه دیگر شبیه‌سازی خاتمه می‌یابد. ($finish یک تکلیف در سیستم است.) نمودار زمانی که از شبیه‌سازی حاصل می‌گردد در شکل ۳۸-۳ نشان داده شده است. زمان کل شبیه‌سازی 200ns طول می‌کشد. ورودی‌های A، B و C پس از 100ns، از 0 به 1 تغییر می‌یابند. در اولین 10ns، خروجی y غیرمشخص است، خروجی x هم در اولین 30ns نامعین می‌باشد. خروجی y پس از 110ns از 1 به 0 می‌رود. خروجی x پس از 130ns از 1 به 0 می‌رود و در 150ns به 1 باز می‌گردد که این مقادیر دقیقاً در جدول ۶-۳ پیش‌بینی شده بود.

عبارات بولی

عبارات بولی در Verilog HDL با عبارت تخصیص مداوم یا پیوسته متشکل از کلمه کلیدی assign که پس از آن یک عبارت بولی آمده مشخص می‌گردد. برای تفکیک علامت جمع حسابی از علامت OR ، Verilog HDL از سمبل (&) ، (|) و (~) به ترتیب برای AND ، OR و NOT استفاده می‌کند. بنابراین برای توصیف مدار ساده شکل ۳۷-۳ با یک عبارت بولی عبارت زیر را به کار می‌بریم.

```
assign x = (A & B) | ~C);
```

مثال ۴-۳ HDL توصیف مداری که با دو عبارت بولی زیر بیان شده را نشان می‌دهد:

$$x = A + BC + B'D$$

$$y = B'C + BC'D'$$

مدار دارای دو خروجی x و y و چهار ورودی A، B، C و D است. دو عبارت assign معادلات بول را توصیف می‌نمایند.

دیدیم که یک مدار دیجیتال می‌تواند با عبارات HDL درست مثل ترسیم در یک نمودار مداری، یا با عبارات بولی توصیف گردد. مزیت HDL این است که برای پردازش به وسیله کامپیوتر مناسب است.

مثال ۴-۳، HDL

```
//Circuit specified with Boolean expressions
module circuit_bln (x,y,A,B,C,D);
   input A,B,C,D;
   output x,y;
   assign x = A | (B & C) | (~B & D);
   assign y = (~B & C) | (B & ~C & ~D);
endmodule
```

مواردی که به وسیله کاربر تعریف می‌شود (UDP)

گیت‌های به کار رفته در توصیف‌های HDL ، با لغات کلیدی and ، or و غیره به وسیله سیستم تعریف می‌شوند و primitives سیستم نام‌گذاری می‌گردند. کاربر می‌تواند primitive های دیگری را با تعریف آنها به صورت جدول اضافه نماید. این نوع مدارها را تعریف شده بوسیله کاربر یا user–defind می‌نامند. یکی از راه‌های معرفی مدار به فرم جدول، معرفی آن با جدول درستی است. توصیف‌های UDP از کلمه کلیدی module استفاده نمی‌کنند. در عوض با کلمه کلیدی primitive (اصلی) تعریف می‌شوند. بهترین راه معرفی primitive ارائه یک مثال می‌باشد.

مثال ۵-۳ HDL یک UDP را با یک جدول درستی تعریف می‌کند. حل آن براساس قواعد کلی زیر است:

مثال ۵-۳، HDL

```
//User defined primitive(UDP)
primitive crctp (x,A,B,C);
   output x;
   input A,B,C;
//Truth table for x(A,B,C) = Σ(0,2,4,6,7)
   table
//      A   B   C  :  x   (Note that this is only a comment)
        0   0   0  :  1;
        0   0   1  :  0;
        0   1   0  :  1;
        0   1   1  :  0;
        1   0   0  :  1;
        1   0   1  :  0;
        1   1   0  :  1;
        1   1   1  :  1;
   endtable
endprimitive

//Instantiate primitive
module declare_crctp;
  reg x,y,z;
  wire w;
  crctp (w,x,y,z);
endmodule
```

- از کلمه کلیدی primitive استفاده شده و به دنبال آن یک نام و لیست پورت‌ها آورده می‌شود.
- تنها یک خروجی می‌تواند وجود داشته باشد که با بکارگیری کلمه کلیدی output و قبل از همه در لیست پورت اعلام می‌شود.
- به هر تعداد ورودی (input) می‌تواند تعریف شود. ترتیب معرفی آنها با اعلام input با ترتیب مقادیرشان در جدولی که به دنبال می‌آید، باید همخوانی داشته باشد.
- جدول درستی باید در داخل کلمات کلیدی table و endtable محصور شود.
- مقادیر ورودی با (:) پایان می‌یابند. خروجی همواره آخرین وارده در هر سطر است و بعد از آن (;) می‌آید.
- در پایان endprimitive ذکر می‌شود.

توجه کنید که متغیرهای لیست شده در بالای جدول بخشی از توضیحات بوده و به منظور آشنایی ذکر شده‌اند. سیستم متغیرها را به ترتیبی که در بخش ورودی ذکر شده‌اند تشخیص می‌دهد. یک UDP نیز مثل primitive سیستم به کار گرفته می‌شود. مثلاً

```
crctp (w,x,y,z)
```

مداری با تابع

$$w(x, y, z) = \Sigma(0, 2, 4, 6, 7)$$

و ورودی‌های x و y و z و خروجی w را پیاده می‌کند.

گرچه Verilog HDL این نوع توصیف را فقط برای UDP به کار می‌برد. دیگر HDL ها و سیستم‌های طراحی کامپیوتری (CAD) روال‌های دیگری را برای مشخص کردن مدارهای دیجیتال به صورت جدول استفاده می‌کنند. جداول می‌توانند با نرم‌افزار CAD برای بدست آوردن یک ساختار گیتی بهینه پردازش شوند.

در این بخش، HDL را معرفی و مثال‌های ساده‌ای از مدل‌سازی ساخت یافته را ارائه دادیم. مشروح مطالب فوق را برای Verilog HDL می‌توان در فصل بعدی یافت. خوانندگانی که با مدارهای ترکیبی آشنا هستند می‌توانند مستقیماً به بخش ۱۱-۴ مراجعه کرده و این موضوع را ادامه دهند.

مسائل

۳-۱ توابع بولی زیر را با استفاده از نقشه سه متغیره ساده نمایید.

(الف) $F(x, y, z) = \Sigma(0, 2, 6, 7)$ (ب) $F(A, B, C) = \Sigma(0, 2, 3, 4, 6)$

(پ) $F(a, b, c) = \Sigma(0, 1, 2, 3, 7)$ (ت) $F(x, y, z) = \Sigma S(3, 5, 6, 7)$

۳-۲ توابع بولی زیر را با استفاده از نقشه سه متغیره ساده کنید.

(الف) $F(x, y, z) = \Sigma(0, 1, 5, 7)$ (ب) $F(x, y, z) = \tilde{\Sigma}(1, 2, 3, 6, 7)$

۳-۳ توابع بولی زیر را با استفاده از نقشه سه متغیره ساده نمایید.

(الف) $xy + x'y'z' + x'yz'$ (ب) $x'y' + yz + x'yz'$

(پ) $A'B + BC' + B'C'$

۳-۴ توابع بولی زیر را با استفاده از نقشه‌های چند متغیره ساده کنید.

(الف) $F(x, y, z) = \sum (2, 3, 6, 7)$

(ب) $F(A, B, C, D) = \sum(4, 6, 7, 15)$

(پ) $F(A, B, C, D) = \sum (3, 7, 11, 13, 14, 15)$

(ت) $F(w, x, y, z) = \sum(2, 3, 12, 13, 14, 15)$

۳-۵ با استفاده از نقشه چهار متغیره توابع زیر را ساده نمایید.

(الف) $F(w, x, y, z) = \sum(1, 4, 5, 6, 12, 14, 15)$

(ب) $F(A, B, C, D) = \sum(0, 1, 2, 4, 5, 7, 11, 15)$

(پ) $F(w, x, y, z) = \sum(2, 3, 10, 11, 12, 13, 14, 15)$

(ت) $F(A, B, C, D) = \sum(0, 2, 4, 5, 6, 7, 8, 10, 13, 15)$

۳-۶ با استفاده از نقشه‌های چهار متغیره توابع زیر را ساده کنید.

(الف) $A'B'C'D' + AC'D' + B'CD' + A'BCD + BC'D$

(ب) $x'z + w'xy' + w(x'y + xy')$

۳-۷ عبارات بولی زیر را با نقشه‌های چهار متغیره ساده کنید.

(الف) $w'z + xz + x'y + wx'z$

(ب) $B'D + A'BC' + AB'C + ABC'$

(پ) $AB'C + B'C'D' + BCD + ACD' + A'B'C + A'BC'D$

(ت) $wxy + yz + xy'z + x'y$

۳-۸ با رسم هر تابع در یک نقشه کارنو، مینترم‌های عبارات بولی زیر را بدست آورید.

(الف) $xy + yz + xy'z$

(ب) $C'D + ABC' + ABD' + A'B'D$

(پ) $wxy + x'z' + w'xz$

۳-۹ موجب‌های اصلی عبارات بولی زیر را بدست آورید.

(الف) $F(w, x, y, z) = \sum(0, 2, 4, 5, 6, 7, 8, 10, 13, 15)$

(ب) $F(A, B, C, D) = \sum(0, 2, 3, 5, 7, 8, 10, 11, 14, 15)$

(پ) $F(A, B, C, D) = \sum(1, 3, 4, 5, 10, 11, 12, 13, 14, 15)$

۳-۱۰ با یافتن موجب‌های اصلی اساسی توابع بول زیر را ساده کنید.

(الف) $F(w, x, y, z) = \sum(0, 2, 4, 5, 6, 7, 8, 10, 13, 15)$

(ب) $F(A, B, C, D) = \sum(0, 2, 3, 5, 7, 8, 10, 11, 14, 15)$

(پ) $F(A, B, C, D) = \sum(1, 3, 4, 5, 10, 11, 12, 13, 14, 15)$

۳-۱۱ توابع بولی زیر را با نقشه‌های پنج متغیره ساده کنید.

(الف) $F(A, B, C, D, E) = \sum(0, 1, 4, 5, 16, 17, 21, 25, 29)$

(ب) $F = A'B'CE' + A'B'C'D' + B'D'E' + B'CD' + CDE' + BDE'$

۳-۱۲ توابع بولی زیر را به صورت ضرب حاصل جمع‌ها ساده کنید.

(الف) $F(w, x, y, z) = \sum (0, 2, 5, 6, 7, 8, 10)$ (ب) $F(A, B, C, D) = \prod(1, 3, 5, 7, 13, 15)$

۳-۱۳ عبارات بولی زیر را به صورت (۱) جمع حاصلضرب‌ها و (۲) ضرب حاصل جمع‌ها ساده کنید.

(الف) $x'z' + y'z' + yz' + xy$

(ب) $AC' + B'D + A'CD + ABCD$

(پ) $(A' + B' + D')(A + B' + C')(A' + B + D')(B + C' + D')$

۳-۱۴ به سه طریق تابع بولی زیر را با هشت لیترال یا کمتر نشان دهید:

$$F = A'B'D' + AB'CD' + A'BD + ABC'D$$

۳-۱۵ تابع بولی F زیر را با حالات بی‌اهمیت d ساده کرده و سپس تابع ساده شده را به صورت جمع مینترم‌ها در آورید:

(الف) $F(x, y, z) = \sum (0, 1, 2, 4, 5)$, $d(x, y, z) = \sum (3, 6, 7)$

(ب) $F(A, B, C, D) = \sum(0, 6, 8, 13, 14)$, $d(A, B, C, D) = \sum(2, 4, 10)$

(پ) $F(A, B, C, D) = \sum(1, 3, 5, 7, 9, 15)$, $d(A, B, C, D) = \sum(4, 6, 12, 13)$

۳-۱۶ عبارات زیر را ساده کنید و آنها را با مدارات دو طبقه گیت NAND پیاده‌سازی نمایید:

(الف) $AB' + ABD + ABD' + A'C'D' + A'BC'$

(ب) $BD + BCD' + AB'C'D'$

۳-۱۷ یک نمودار گیت NAND رسم کنید به نحوی که تابع زیر را پیاده‌سازی کرده آن را متمم نماید:

$$F(A, B, C, D) = \sum(0, 1, 2, 3, 4, 8, 9, 12)$$

۳-۱۸ یک نمودار منطقی با استفاده از گیت‌های NAND دو ورودی برای پیاده‌سازی عبارت زیر رسم کنید.

$$(AB + A'B')(CD' + C'D)$$

۳-۱۹ توابع زیر را ساده کنید و آنها را با مدارهای دو طبقه گیت NOR پیاده نمایید:

(الف) $F = wx' + y'z' + w'yz'$ (ب) $F(w, x, y, z) = \sum(5, 6, 9, 10)$

۳-۲۰ یک مدار NAND چند طبقه برای عبارت زیر رسم کنید.

$$(AB' + CD')E + BC(A + B)$$

۳-۲۱ یک مدار چند طبقه NOR برای عبارت زیر رسم کنید.

$$w(x + y + z) + xyz$$

۳-۲۲ نمودار مدار شکل ۴-۴ را به مدار چند طبقه NAND تبدیل نمایید.

۳-۲۳ تابع بول F را همراه با حالات بی‌اهمیت d با کمتر از دو گیت NOR پیاده‌سازی نمایید.

$$F(A, B, C, D) = \sum(0, 1, 2, 9, 11)$$
$$d(A, B, C, D) = \sum(8, 10, 14, 15)$$

فرض کنید که هر دو نوع ورودی معمولی و متمم موجودند.

۳-۲۴ تابع بول F را با فرم‌های دو طبقه (الف) NAND–AND، (ب) AND–NOR، (پ) OR–NAND و (ت) NOR–OR پیاده‌سازی کنید.

$$F(A, B, C, D) = \Sigma(0, 1, 2, 3, 4, 8, 9, 12)$$

۳-۲۵ هشت فرم دو طبقه غیرمفید (زاید) نمایش تابع را لیست کنید و نشان دهید که به یک عمل کاهش می‌یابند. نشان دهید که این فرم‌ها چگونه برای گسترش تعداد ورودی‌ها به کار می‌روند.

۳-۲۶ با استفاده از نقشه‌ها، ساده‌ترین فرم جمع حاصلضرب تابع F = fg را بیابید، که در آن f و g برابرند با

$$f = wxy' + y'z + w'yz' + x'yz'$$

و

$$g = (w + x + y' + z')(x' + y' + z)(w' + y + z')$$

۳-۲۷ نشان دهید که دوگان XOR، متمم آن هم هست.

۳-۲۸ با توازن فرد، مدارهای مولد توازن سه بیتی و چک‌کننده توازن چهار بیتی را بدست آورید.

۳-۲۹ عبارات بولی زیر را با نیم جمع‌کننده پیاده‌سازی کنید.

$$D = A \oplus B \oplus C$$
$$E = A'BC + AB'C$$
$$F = ABC' + (A' + B')C$$
$$G = ABC$$

۳-۳۰ عبارت بولی زیر را با گیت‌های XOR و AND پیاده کنید.

$$F = AB'CD' + A'BCD' + AB'C'D + A'BC'D$$

۳-۳۱ توصیف ساختار گیتی HDL مدار شکل ۳-۲۲ (الف) را بنویسید.

۳-۳۲ مدار XOR شکل ۳-۲۲ (الف) دارای گیت‌هایی با تأخیر 10ns برای هر وارونگر، 20ns تأخیر برای هر گیت AND و 30ns برای هر گیت OR است. ورودی مدار از xy = 00 به xy = 01 می‌رود.
(الف) سیگنال‌ها را در خروجی هر گیت از t = 0 تا t = 50ns معین کنید.
(ب) توصیف HDL را برای مدار با در نظر گرفتن تأخیرها بنویسید.
(پ) یک مدول محرک (مشابه مثال ۳-۳ HDL) بنویسید و مدار را برای تأیید پاسخ بخش (الف) شبیه‌سازی کنید.

۳-۳۳ توصیف HDL مدار شکل ۳-۳۷ را با دو عبارت بنویسید.

۳-۳۴ توصیف HDL مداری را که با توابع بولی زیر مشخص شده بنویسید. از عبارات تخصیص یافته پیوسته استفاده نمایید.

$$x = A(CD + B) + BC'$$
$$y = (AB' + A'B)(C + D')$$
$$z = [(A + B)(C' + D'B)]'$$