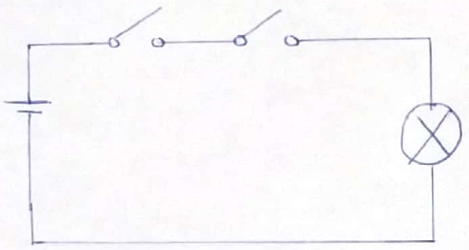
PLD 2

۱- ص ۲ :

- برای پیاده‌سازی مدارها نیازمند یک سری قطعات مثل کلید هستیم که ترانزیستور می باشند.

- مفهوم AND :

A	B	F
۰	۰	۰
۰	۱	۰
۱	۰	۰
۱	۱	۱

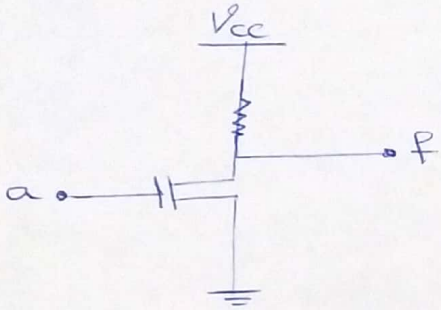


زمانی لامپ روشن می شود که هردو کلید بسته باشند.

ترانزیستور ← کلیدی که به صورت الکترونیکی قابل کنترل باشد.

- اگر بتوانیم ترانزیستور را بسازیم می توانیم همه قطعات و گیت ها را با هاش بسازیم.

- ساختار گیت Not با ترانزیستور :



a	F
۰	۱
۱	۰

- discrete ← پیاده‌سازی مدار با استفاده از قطعات جدا از هم (برای مثال برای ساخت گیت NAND ۴ تا ترانزیستور جدا از هم درست می کنیم بعد با سیم وصل می کنیم)

- IC ← مجموعه‌ای از قطعات الکترونیکی که با تعداد زیاد مثل پکیج در کنار هم قرار دارند و سیم بندی شده اند.

تا اسلاید ۳

۲- ص ۵ :

- ابزارهای سنتز ← نرم افزاری است که می تواند یک کد HDL را به یک مدار تبدیل کند.

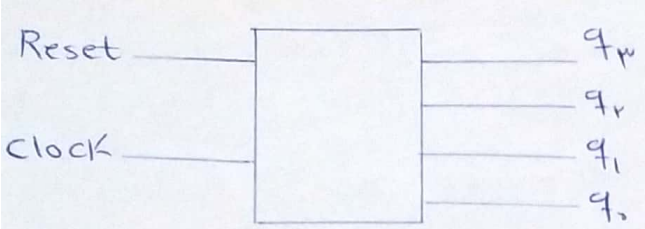
۳- ص ۶ : طراحی باید به صورت سلسله مراتبی باشد برای مثال در CPU :

ابتدا بلوک دیاگرام هایی برای CPU در نظر می گیریم که یک بخش ALU یک بخش RAM و IR و ..... دارد و بعد بیاییم داخل هرکدام از این ها را بخش بندی کنیم تا در نهایت به کوچکترین مرحله که گیت ها هستند برسیم.

- هرچقدر تعداد بخش ها و گیت ها بالا رود پیچیدگی مدار بالا می رود و هرچقدر پیچیدگی بالا رود تحلیل و طراحی مدار سخت تر می شود و زمان برتر. به همین خاطر برای طراحی مدارهای پیچیده از ساختار سلسله مراتبی استفاده می کنیم.

Scanned by CamScanner

۴- ص ۷:



- بلوک دیاگرام شمارنده‌ی حلقوی:

مدار داخلی اش از ۴ تا فلیپ فلاپ T ساخته شده است.

فلیپ فلاپ T هم از فلیپ فلاپ D و یک گیت Not

- شمارنده حلقوی متشکل از فلیپ فلاپ T:

شمارنده‌ای است که وقتی سیگنال Reset غیرفعال باشد و بهش کلاک بدهیم یک دسی اعداد را از ۰ تا ۴ برای ما تولید می کند.

۵- ص ۸:

- Leaf cell ← ساده‌ترین سطح یا پایین‌ترین سطح

- فرض کنید می خواهیم یک CPU را پیاده سازی کنیم:

ابتدا گیت‌ها را می سازیم. سپس با ترکیب این گیت ها می توانیم مدارات نسبتاً پیچیده را پیاده‌سازی کنیم.

(مثل FA ها) حالا می توانیم FA که یک بیت را جمع می کرد را ۸ تا کنار هم قرار دهیم می توانیم یک جمع کننده ۸ بیتی بسازیم و یک دسی مدارات دیگر بهش اضافه می کنیم و ...

۶- ص ۹:

- در این روش مثلا برای طراحی CPU قسمت های مختلف آن را به صورت بلوک می بینیم. (بلوک ALU و .....) بعد یک مرحله پایین می آییم و بر داخل ALU تمرکز می کنیم که بخش های منطق، حسابی و ... دارد و بعد باز بخش حسابی عملیات های + و - و x و / را باید داشته باشد و بعد برای + باید مدار جمع کننده داشته باشیم و .....

غیرهم جنس

- برای سیستم هایی که پیچیدگی بالایی دارند و بلوک هایشان نامتناجس است از روش بالا به پایین استفاده می کنیم.

- Register transfer level (RTL):

در معماری کامپیوتر یک دسی چیزهایی به شکل روبرو داشتیم:

$$\begin{cases} AC \leftarrow AC + DR \\ R_4 \leftarrow R_2 + R_3 \end{cases}$$

R_4 یک رجیستر ۸ بیتی است ولی در این جا مثل یک متغیر عمل می کنیم و یک دسی عملگر + داریم ولی به جای آپکد بیاییم و مدارش را بکشیم با سخت افزار مثل یک متغیر برخورد می کنیم که بهش RTL گویند.

سنتز ← تبدیل روابط RTL به مدار یا بلوک های مداری

سه شنبه ۱۴۰۲/۰۷/۲۵

جلسه‌ی ۳

۷- ص ۲: برای این که HDL Verilog بتواند سلسله مراتب طراحی را پشتیبانی کند، ۴ سطح از توصیف مدار را در اختیار ما قرار می دهد.

Scanned by CamScanner

not (ورودی و خروجی) and (ورودی ۲ و ورودی ۱ و خروجی) OR (ورودی ۲ و ورودی ۱ و خروجی)

۸- ص ۳ :
- در این توصیف وارد بحث گره ها و لیت ها نمی شویم و سعی می کنیم جریان داده را توصیف کنیم .

- سطح توصیف RTL یا (register transfer level) Data Flow

- برخلاف توصیف ساختاری که بیشتر از لیت ها استفاده می کردیم این جا بیشتر از عملگر استفاده می کنیم و می توانیم مدار را براساس یک سری عملگرها و سیگنال ها توصیف کنیم .

- حجم کد تولیدشده در این روش کمتر از روش قبل است و در این روش نیازی به دستاویز نیست و از روی کد می توانیم تشخیص دهیم .

۹- ص ۴ : این سطح برای توصیف بلوک های منطقی ترکیبی و مدارهای پیچیده به کار می رود و هدف همین راه اجرای رفتار و عملکرد مدار است بدون این که وارد جزئیات طراحی بشویم .

۱۰- ص ۵ : پایین ترین سطح توصیف است و می توانیم رفتار مدار را براساس سوئیچ ها یا ترانزیستورها بیان کنیم

۱۱- ص ۸ :
- ماجول ها را می توان داخل هم هم استفاده کرد .

۱۲- ص ۱۰ :
- در اسلایدهای قبلی TFF را با ماجول پیاده سازی کردیم . اما وقتی می خواهیم در ساختار سطح بالاتر از این ماجول استفاده کنیم باید نمونه هایی از ماجولی که توصیف کردیم را در ماجول دیگر استفاده کنیم .

- برای مثال در شمارنده حلقوی ما یک ماجول کلی داریم که داخلش ۴ تا فلیپ فلاپ T دارد . برای پیاده سازی آن مجبور هستیم ۴ بار از ماجول TFF که قبلا توصیف کرده بودیم ، استفاده کنیم . به این کار نمونه سازی یا instantiation گویند.

(ورودی ها و خروجی ها) + (اسم instance) تک id + اسم ماجولی که می خواهیم استفاده کنیم

۱۳- ص ۱۱ :
- برای اجرای شبیه سازی و این که آیا درست کار می کند یا نه از بلوکی به اسم بلوک تحریک یا test bench یا stimulus استفاده می کنیم که یک بلوکی است که می تواند یک سری داده هایی را تولید کند و در اختیار بلوک طراحی قرار دهد .

- بلوک stimulus ، سیگنال هایی را که برای کارکرد مدار لازم است را در اختیار قرار می دهد .
مثلا در شمارنده حلقوی ما می آییم و از طریق یک بلوکی (بلوک تحریک) کلاک آن را تولید می کنیم و یک سیگنال ریست هم بهش می دهیم در این حالت انتظار داریم که کار کند و خروجی های مدار را هم به بلوک تحریک وصل می کنیم و بعد شبیه سازی را انجام می دهیم و بلوک تحریک سیگنال های مورد نیاز را طبق چیزی که نوشته شده

Scanned by CamScanner

بجش اعمال می کند ورودی ها را در محیط شبیه سازی نشان می دهد.

۱۴-

test bench خود نیز یک ماجول است و معمولا سعی می شود آن را خارج از طراحی در نظر بگیریم.
(یک ماجول کلی و ۲ تا ماجول مجزا از هم)

- ماجول test bench صرفا برای شبیه سازی استفاده می شود.

PLD 4

۱۵- حل:

۸'hAA ← یک عدد ۸ بیتی در مبنای هگز

- اگر مبنای عدد را مشخص نکنیم پیش فرض هگز در نظر گرفته می گیرد. مثل ← ۸'F1

۱۶- حل:

- اندازه کوچکتر از مقدار داده شده مثلا

۶'hFF ← ۳F = xx11 1111 (۳F)

۸ بیت ۶ بیت

عدد پیاده سازی شده

- ~ بزرگتر ~ ~ ; مثلا

۱۰'hFF ← 00 1111 1111

۸ بیت ۱۰ بیت

Verilog آخرین عدد نوشته شده مقدار را که FF است را چک می کند

۱۰۰۰۰ یا ۰۰۰۰۰ ← با صفر پر می کند.

x ← x

z ← z

- در Verilog ما یک منطق ۴ وضعیتی داریم (یک بیت ۴ وضعیت می تواند داشته باشد) که

۰ و ۱ و x و z هستند.

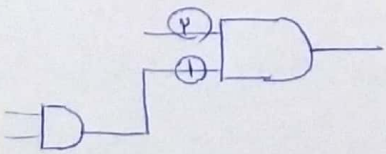
نامشخص های امپدانس

مثال: ۴'bx1 ← xxx1

۱۷- حل:

- هنگام تعریف طیف طاب، لچ و رجیستر و... از نوع داده ی Reg استفاده می کنیم.

- فرض کنید یک مداری به شکل زیر داریم:



ورودی ① مقدارش مشخص است و اگر از لیت قبلی ه بیاید می شود ابیاید امی شود.

اما ② که به هیچ جا وصل نیست اصطلاحا بهش امپدانس بالا می گویند. (پین ورودی که به هیچ جایی وصل نیست)

Scanned by CamScanner