Subject: Year: Month: Date:

خلاصه بخش دوم فصل ۴

در مورد مدارهای NAND میدانیم میتوان برخی از توابع غیر پیش پا افتاده را محاسبه نمایند در ادامه خواهیم دید میتوانیم تمام توابع متناهی را با مدارات NAND محاسبه نماییم

قضیه universal بودن گیت NAND: برای هر تابعی به شکل

$f:\{0,1\}^n \longrightarrow \{0,1\}^m$ و $m,n>0$

و اگر مقداری مانند $c>0$ وجود داشته باشد میتوانیم بگوییم NAND circ-prog وجود دارد که حداکثر $cm2^n$ خط میتواند این تابع را محاسبه کند

این قضیه برای NAND circ programها گفته شده است ولی از آنجایی که این برنامه‌ها با مدارات NAND یکسان هستند میتوانیم آنها را به جای یکدیگر نیز استفاده کنیم

قضیه ۴.۱۳ یونیورسال بودن مدارهای بولی: اگر $m,n>0$ باشند و ثابتی مثبت مانند c وجود داشته باشد میتوانیم بگوییم میتوانیم توابعی به فرم $f:\{0,1\}^n\to\{0,1\}^m$ را با حداکثر $cm2^n$ گیت پیاده‌سازی نماییم

بنابراین از این قضیه متوجه میشویم تمامی تابع‌های متناهی در نهایت با مدارهای بولی قابل پیاده‌سازی هستند

Subject: Year: Month: Date:

پس برای اینکه این NAND CIRC Program را به LookUp Function

تبدیل کنیم ابتدا 2^n تا متغیر ورودی برای حالت های مختلف n بیت

ورودی داریم بعد هرکدام از این خط ها این 2^n خط برنامه میتوانند دو مقدار

۰ و ۱ داشته باشند که برای آنکه این ۰ و ۱ ها را با NAND پیاده سازی کنیم

۱- 1 = NAND(zero, zero)

۲- 0 = NAND(1, 1)

۳ خط نیاز داریم:

۳- برای زمانی که یکی از تعریف ها وجود ندارد یا مقدار undefined باشد ?*

پس برای مقدار دهی اولیه به $2^n + 3$ خط برنامه نیاز داریم

همچنین اثبات کردیم برای آنکه LookUp Func بتواند تعمیم خود را به ورودی

را حساب کند به $4(2^n)$ خط نیاز دارد پس در کل $4(2^n) + 2^n + 3$

خط نیاز داریم پس حداکثر ثابت نمودیم با $c\,m\,2^n$ خط میتوانیم

هر مداری را با NAND CIRC Program

این نتیجه را میتوانیم بهبود بدیم و آن را به حداکثر $c\,m\,\frac{2^n}{n}$ خط کاهش دهیم

در این قضیه c حداکثر ۱۰ است و میتواند به صورت دلخواه نزدیک ۱ باشد

Subject: Year: Month: Date:

اثبات یونیورسال بودن گیت NAND : به صورت کلی از آنجایی که تمامی مدارهای

بولی از سه گیت پایه AND و OR و NOT ساخته شده اند پس اگر بتوانیم نشان دهیم

که NAND میتوانیم آنها را با آن بسازیم ثابت نموده ایم NAND یونیورسال است اما

از آنجایی که در قضیه گزاره بالایی برای تعداد گیت های NAND تعریف کرده است

اثبات آن به روش زیر میباشد

برای اثبات فرض میکنیم $m=1$ باشد زیرا در این صورت خروجی نهایی تابع ما

تک بیتی خواهد بود بنابراین $F: \{0,1\}^n \to \{0,1\}$ یعنی تابع n بیت ورودی و یک

بیت خروجی دارد برای هر تابع میتوانیم جدول درستی آن را ایجاد نماییم به عنوان

مثال :

a	b	$a \vee b = f$
0	0	0
0	1	1
1	0	1
1	1	1

ورودی — خروجی

بنابراین اگر n ورودی داشته باشیم جدول 2^n ورودی دارد

این 2^n ورودی را میتوانیم با یک lookup function نشان دهیم

$$G_{00}=0$$
$$G_{10}=1$$
$$G_{01}=1$$
$$G_{11}=1$$
$$a=X[0]$$
$$b=X[1]$$

$$\Rightarrow Y[0] = lookup\_2(G_{00}, G_{10}, G_{01}, G_{11}, X[0], X[1])$$

حال هرکدام از G_{xx} ها را میتوانیم براساس مدارات NAND بنویسیم

$G_{xx}=1 = NAND(zero, zero)$

$G_{xx}=0 = NAND(1, 1)$

Helix

Subject: Year: Month: Date:

اثبات دیگر برای قضیه ۴.۱۴ : این اثبات کران بهتری نیست

به اثبات قبلی دارد و بنابراین با تعداد بیشتری گیت مدارها را پیاده‌سازی می‌کند اما روشی اثبات ساده‌تری دارد

قضیه ۴.۱۷ : به‌ازای هر m, n تابعی مانند $f: \{0,1\}^n \to \{0,1\}^m$ وجود دارد که با استفاده از آن نمی‌توانیم هر تابعی را با مداری بولی با حداکثر $cmn2^n$ گیت محاسبه نماییم

مانند روشی قبل نیز ابتدا فرض می‌کنیم $m=1$ می‌باشد خروجی تک‌بیتی است

α یعنی از ورودی است

می‌گوییم برای هر $\alpha \in \{0,1\}^n$ یک مدار می‌گوییم $O(n)$ وجود دارد

که تابع مقابل را محاسبه نماید $\delta_\alpha : \{0,1\}^n \to \{0,1\}$

مقدار خروجی این تابع به صورت زیر است $\delta_\alpha(x) = 1$ اگر و تنها اگر $x = \alpha$

یعنی خروجی α که برای تمام مقادیر x صفر است به جز $x = \alpha$ حال با استفاده از این توابع ما می‌توانیم هر تابعی را به صورت حداکثر $n2^n$ تابع δ_α نوشت

بدست بیاوریم ★ نوشتن یک مدار بولی به صورت جمع ★ Minterm

Helix

Subject: Year: Month: Date:

برای اینترم ها داشتیم هر جدول درستی را میتوانیم مثل جدول

قضیه قبل را به صورت $ba' + b'a + ab = a \vee b$

بنویسیم حال با این ایده باید به اثبات این قضیه بپردازیم بیشتر مثل

تعداد کیت ها هم نسبت به روش های قبلی آن است که مدارای که با این

صورت نوشته میشوند قابل ساده سازی باشند با استفاده از مثلاً جدول کارنو

ما میتوانیم هر مداری که از ترکیب N تا OR دوورودی تشکیل شده اند

(N two-input OR)

بنابراین اگر مداری با یک $O(N)$ برای هر سه گیت داشته باشیم

برای همه $x \in \{0,1\}^n$ باشد ما میتوانیم f را برای هر گیت مداری به سایز $O(nN)$

$O(nN) = O(n2^n)$

بنابراین برای اثبات آن باید ثابت کنیم برای هر تابع f یک مداری

با اندازه $O(n2^n)$ وجود دارد که f را محاسبه کند

اول با فرض میگیریم $\delta_\alpha(x) = \begin{cases} 1 & x = \alpha \\ 0 & \text{o.w.} \end{cases}$

و همچنین مداری با حداکثر $2N$ گیت وجود دارد تا δ_α را محاسبه کند

اثبات این بخش را با یک مثال آن زدیم

$\delta_{011} : \{0,1\}^3 \rightarrow \{0,1\}$

بنابراین فرض $\delta_{011} = \bar{x}_0 \wedge x_1 \wedge x_2$

هر تابع و یا جدول درستی را میتوانیم به صورت جمع یا OR δ_a ها بنویسیم

بنابراین $f(x) = \delta_{x_0}(x) \vee \delta_{x_1}(x) \vee \dots \vee \delta_{x_{N-1}}$

در بدترین حالت $\delta_{000} = \bar{x}_0 \wedge \bar{x}_1 \wedge \bar{x}_2$

مانند حالت مقابل برای هر δ_a ها n تا گیت AND میخواهیم

و همچنین n تا گیت NOT زیرا در بدترین حالت تمامی ورودی ها صفر هستند
پس باید آن n ورودی ها را AND کنیم پس حداکثر $2n$ گیت نیاز داریم
برای تک تک δ_a ها حالا در کل چند گیت برای همه نیاز داریم:

بدترین حالت اینجوری است که جدول درستی ما به ازای هر سطر خروجی 1 داشته باشد یعنی

a	b	f
0	0	1
0	1	1
1	0	1
1	1	1

$f(x) = \delta_{x_0} \vee \delta_{x_1} \vee \dots \vee \delta_{x_{N-1}}$

N تا سطر داره جدول درستی n

کمک n δ_a N داریم و همچنین N تا گیت OR

$f = N\delta_a + N$ n بیت ورودی δ_a ، تعداد سطرهای $n = N = 2^n$

$= 2nN + N = 2n(2^n) + 2^n$

Subject: Year: Month: Date:

پس بنابراین تعداد کل گیت های ما $O(n2^n)$

تعریف ۴.۱۶ $SIZE_{n,m}(s)$

تاکنون مشاهده کردیم که میتوانیم تمامی توابع متناهی به فرم

$f:\{0,1\}^n \to \{0,1\}^m$

را با حداکثر $O(m2^n)$ می سازیم برخی از توابع مانند جمع و ضرب

را میتوانیم با تعداد خیلی کمتری گیت پیاده سازی نماییم در این بخش تعریف

ارائه میدهیم با عنوان $SIZE_{n,m}(s)$ که مجموعه ای است از توابع

که n بیت ورودی را به m بیت خروجی نگاشت میکند

و میتوانیم آن ها را با مدارهای NAND با حداکثر s گیت پیاده سازی کنیم

تعریف ۴.۱۸

برای تمام اعداد طبیعی n,m,s تعریف میکنیم

$SIZE_{n,m}(s)$

مجموعه تمامی توابعی است که میتوانند با حداکثر s گیت NAND پیاده سازی

شوند

Helix

Subject: Year: Month: Date:

به این معنی است که اگر یک تابعی دارای $size_{n,m}(s)$ باشد برای

چند مدار NAND وجود دارند که میتوانند f را با حداکثر s گیت بسازند

به عنوان مثال:

$size_{2,1}(5) \Rightarrow \{f:\{0,1\}^2 \to \{0,1\}\}$

مجموعه تمام توابعی است که میتوانند با کمتر از ۵ گیت پیاده سازی شوند

$size_n(s)$ برای حالتی است که m=1 است

$size(s)$: مجموعه تمام توابعی است که با هر تعداد اینپوت داده شده

میتوانند با حداکثر s گیت ساخته شوند که اجتماع $size_{n,m}(s)$ است

به ازای مقادیر مختلف n,m که میتوانند حداکثر با s گیت پیاده سازی شوند

یعنی بدون توجه به ورودی و خروجی اگر تابعی را size(s) وجود داشت

با s گیت قابل پیاده سازی است

این ها این تعریف ها برای توابع هستند و برای مدارها و برنامه ها کاربرد ندارند

زیرا در حالی که یک برنامه یک تابع را محاسبه میکند برابر با تابع نیست

تعریف size را برای گیت های AND, OR, NOT هم میتوانیم بیاوریم

Subject: Year: Month: Date:

لم ۴.۱۹: اگر فرض نمائیم و تعریف کنیم $Size^{AON}_{n,m}(s)$ شامل

مجموعه تمام توابعی باشد که با حداکثر s گیت AND, OR, NOT قابل پیاده

سازی شوند داریم: $Size_{n,m}(s/2) \le Size^{AON}_{n,m}(s) \le Size_{n,m}(3s)$

اثبات: اگر f را بتوانیم با حداکثر $\frac{s}{2}$ گیت NAND پیاده سازی کنیم

می‌توانیم هر گیت NAND را با دو گیت AND, NOT جایگزین نمائیم

بنابراین همان مدار را توانستیم با s گیت AND, OR, NOT بسازیم

از طرف دیگر طبق قضیه ۴.۱۲ می‌دانیم s تا گیت AND, OR, NOT را می‌توانیم

با 3s گیت NAND پیاده سازی کنیم

تمرین ۴.۱: اگر فرض کنیم تابع f عضو کلاس $Size_n(s)$ باشد آنگاه نشان دهید مکمل تابع f که تابع g می‌باشد $g(x)=1-f(x)$ آنگاه ثابت نمائیم عضو کلاس مقابل است $1-f \in Size_n(s+c)$

حل: اگر فرض کنیم f از $Size_n(s)$ است پس برنامه‌ای که خطوط آن را می‌سازد

خط آخر برنامه که خروجی بوده را تغییر می‌دهیم $Y[0] \to temp$

و خط آخر جدید را برای تابع g می‌نویسیم $Y[0] = NAND(temp, temp)$

پس با یک خط بیشتر توانستیم تابع g را محاسبه نمائیم