دانشگاه صنعتی اصفهان

دانشکده برق و کامپیوتر

گزارش انجام پروژه درس زبان توصیف سخت افزار و مدارات

پیاده سازی فیلتر FIR ، بهینه سازی آن و استفاده از هسته از پیش طراحی شده آن

استاد مربوطه:

دکتر مجید نبی

دانشجویان :

مهدی مصون 9431293

عماد ملک حسینی 9431553

کیوان نصرتی 9432513

مرتضی موسی پسندی 9431923

خرداد و تیر سال 97

فهرست مطالب

عنوان **صفحه**

1. مقدمه

1-1. هدف پروژه

در این پروژه هدف پیاده سازی فیلتر FIR با استفاده از چند روش مختلف و بررسی خصوصیات فیلتر به دست آمده با استفاده از هر روش و حذف نویز یک فایل صوتی است . ما در این پروژه موظف به پیاده سازی فیلتر FIR و بهبود آن توسط Pipeline و گذاشتن قید های زمانی یا مکانی و استفاده مناسب و بهینه از ضرب یا جمع کننده و سرانجام استفاده از هسته از پیش طراحی شده برنامه Xillinx و تست کردن این سه فاز و اعلام میزان تاخیرها و میزان حجم های مصرفی هر یک از این فاز ها هستیم .

1-2. معرفی نویز

نویز نوعی اختلال در حوزه فرکانسی و زمانی یک سیگنال ا ست. از دیدگاه فیزیک، نویز از صدا قابل تشخیص نیست، زیرا هر دو لرزش از طریق یک رسانه مانند هوا و آب است. این اختلاف زمانی رخ می دهد که مغز صدا را دریافت و درک کند.

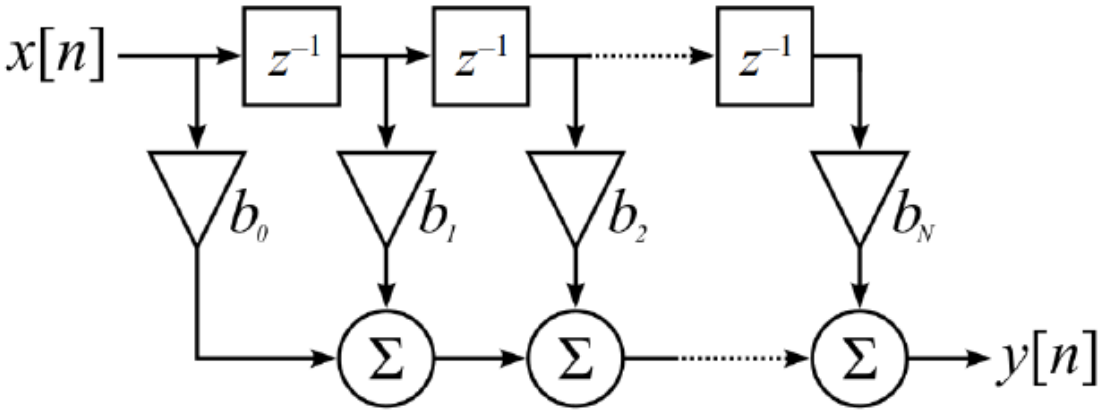
در علوم تجربی، نویز می تواند به هر نوسانات تصادفی داده ها اشاره کند که مانع درک سیگنال مورد انتظار می شود.حال چگونه نویز را از سیگنال ورودی خود حذف کنیم؟!

1-3. معرفی فیلتر

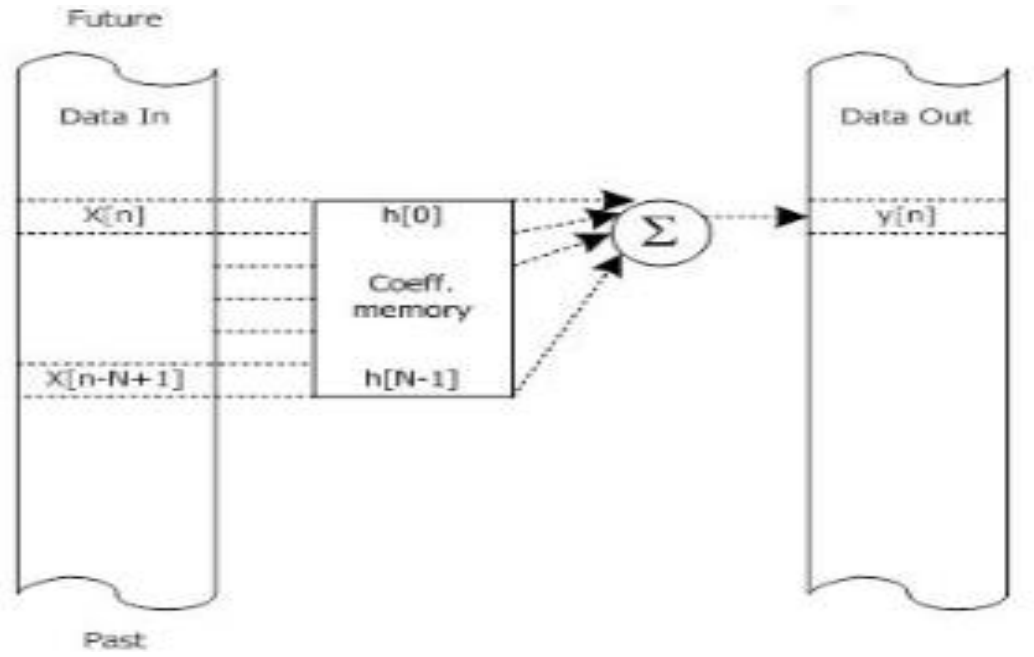
فیلتر به ابزاری گفته می شود که میتواند نویز یا عامل ناخواسته را حذف کند. ما چندین نوع فیلتر داریم. میتوان به صورت های خطی و غیر خطی ، دیجیتال و آنالوگ ، تغییر پذیر یا ناپذیر با زمان ، علی یا غیر علی و پاسخ بی نهایت ضربه یا پاسخ تعداد محدودی ضربه تقسیم بندی کرد.

4-1. بلوک دیاگرام فیلتر های FIR

FIR به معنای پاسخ ضربه محدود است . در فیلتر FIRتقریبا هیچ باز خوردی نداریم. مزیت این فیلتر ها نسبت به IIR این است که میتوانند به راحتی با فاز خطی طراحی شود به عبارت دیگر فیلتر فاز خطی سیگنال را به تاخیر می اندازد اما فاز آن دارای تحریف نیست.



شکل ۱ - نمای کلی یک فیلتر FIR در پیاده‌سازی دیجیتالی



دیاگرام کلی FIR 1

2. محاسبات برای فیلتر FIR

فرمول ساده فیلتر Fir به این صورت است :

$$y[n] = b_0 x[n] + b_1 x[n-1] + \cdots + b_N x[n-N]$$
$$= \sum_{i=0}^{N} b_i \cdot x[n-i],$$

و برای پاسخ ضربه آن نیز به این صورت است :

$$h[n] = \sum_{i=0}^{N} b_i \cdot \delta[n-i] = \begin{cases} b_n & 0 \leq n \leq N \\ 0 & \text{otherwise.} \end{cases}$$

و برای پاسخ فرکانسی هم داریم :

$$H_{2\pi}(\omega) \stackrel{\text{def}}{=} \sum_{n=-\infty}^{\infty} h[n] \cdot \left(e^{i\omega}\right)^{-n} = \sum_{n=0}^{N} b_n \cdot \left(e^{i\omega}\right)^{-n},$$

3. بررسی نتایج فاز ها

3–1 .نتایج فاز یک (طراحی فیلتر FIR) :

با بررسی رابطه ورودی و خروجی فیلتر FIR به این نتیجه رسیدیم که هر جا در فرمول اندیس منفی مشاهده نمودیم مربوط به ورودی های سابق بوده است که فهمیدیم نیاز به شیفت ریجستر و حافظه داریم . پس ما فهمیدیم نیاز به حلقه داریم. سپس شروع به پیاده سازی آن کردیم به این صورت که ضرایب ورودی را پارامتری گرفته و پس از گرفتن 19 ورودی آنها را در 19 تا خونه 16 بیتی که به عنوان شیفت ریجستر استفاده میکنیم ذخیره میکنیم؛ سپس در یک متغیر 36 بیتی محاسبات مربوط به فیلتر را حساب و سپس 16 بیت پر ارزش را به عنوان خروجی تحویل می دهیم. ما در اینجا از 18 تا جمع کننده 2 ورودی (که در یک کلاک به عنوان یک جمع کننده 19 ورودی عمل میکند) و 19 تا ضرب کننده برای گرفتن خروجی استفاده میکنیم. ما در هر کلاک یه ورودی میگیریم و در همان کلاک خروجی همان ورودی را میگیریم.

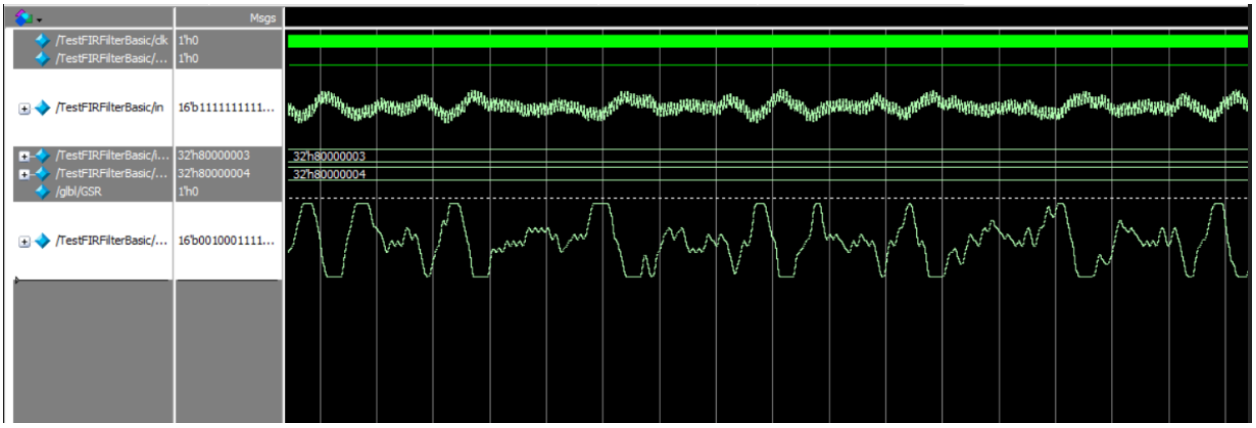
ما در تست بنچ علاوه بر عملیات کلاک دهی فایل ورودی را با in به ماژول داده و فایل خروجی را از آن با out تحویل میگیریم.

صدای خروجی تحویل گرفته شده تقریبا مطلوب است و نویز مشاهده شده در آن به علت امکان overflow شدن دیتا بعد از ضرب کردن و جمع کردن در آن متغیر مذکور باشد.

میزان فضای حافظه استفاده شده بر روی FPGA : 2%

میزان تاخیر : 24.2 ns

میزان فرکانس : 41.3MHz



شکل موج فاز 1

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	325	11440	2%
Number of Slice LUTs	2397	5720	41%
Number of fully used LUT-FF pairs	79	2643	2%
Number of bonded IOBs	34	102	33%
Number of BUFG/BUFGCTRLs	1	16	6%
Number of DSP48A1s	1	16	6%

خلاصه طراحی فاز 1

3-2. نتایج فاز دو (بهینه سازی فیلتر FIR)

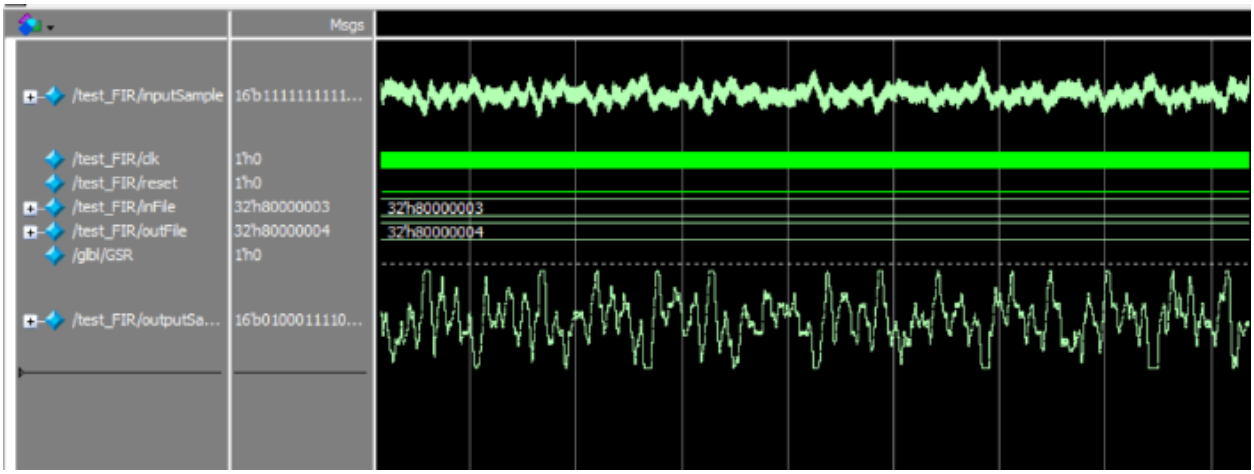
ما مستحضریم که در resource sharing علاوه بر کاهش میزان حافظه استفاده شده سرعت هم کاهش میابد. حال میدانیم در قسمت pipeline تاخیر کم شده و فرکانس کاری افزایش میابد ولی نسبت به resource sharing فضای بیشتری ذخیره میکند.

حال با پخش کردن جمع در پنج مرحله و ضرب در یک مرحله pipeline انجام دادیم ما برای اینکار از 19 تا ضرب کننده و 16 تا جمع کننده دو ورودی استفاده کردیم . به این صورت که در 7 تا استیج کل کار انجام میشود. در استیج اول کلیه ضرب ها انجام میشود و در پنج لایه بعدی جمع انجام شده و در لایه آخر هم خروجی نوشته میگردد.

میزان فضا حافظه استفاده شده بر روی FPGA : 8٪

میزان تاخیر : 5ns

میزان فرکانس : 197MHz



شکل موج فاز 2

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	988	11440	8%
Number of Slice LUTs	1319	5720	23%
Number of fully used LUT-FF pairs	934	1373	68%
Number of bonded IOBs	34	102	33%
Number of BUFG/BUFGCTRLs	1	16	6%
Number of DSP48A1s	16	16	100%

خلاصه طراحی فاز 2

3-3. نتایج فاز سه (استفاده از هسته ی از پیش طراحی شده ی FIR Compiler)

نرم افزار Xillinx ه سته هایی رایگان در اختیار م شتریان قرار داده ا ست تا آنها بتوانند کد های خود را از لحاظ کارایی و کمیت و کیفیت با آنها مقایســـه کنیم و اینکه بدون درگیری با طراحی داخلی میتونیم از امکان بهره وری استفاده کنیم.

برای انجام این فاز از پروژه تنها کافی بود که ما هسته FIR Compiler نسخه پنجم را فراخوانی کردیم. فرقی در خروجی ها از لحاظ speed یا area در تنظیم نیست ولی از لحاظ تاخیر و میزان حافظه متفاوت اند که ما میتوانیم بر اساس نیاز مان (سرعت) آنرا تنظیم کنیم و پارامترهای خواسته شده را به آن ارائه دهیم. سپس با استفاده از یک کد تست مشابه همان کد تستی که در فاز اول نوشتیم، داده های ترانه ورودی(تایتانیک با خش) را به آن دادیم و خروجی را روی مدلسیم مشاهده کردیم و مانند تمامی بخش ها آنرا در فایل های appendix ذخیره کردیم.

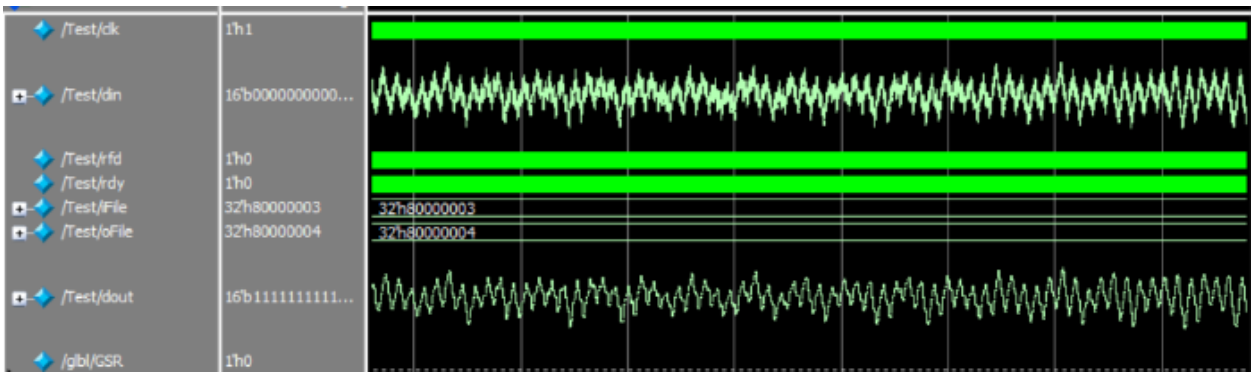
این هسته برای پردازش یک کلاک میگیرد و ورودی sample صوت را با فرکانس دیگری به آن میدهیم. در این مثال همانگونه که گفته شده 50MHZ و فرکانس ورودی sample نیز 44100 سمپل بر ثانیه است.

ضرایب coef ها را با فایل داده شده لود کردیم . یکی از نیازمندی های ما این است که خروجی ما 16 بیت باشد پس 36 بیت حاصل از محاسبات core را round کنیم . خود نرم افزار تنظیمات rounding mode دارد برای map کردن این عدد بزرگ به 16 بیت چند روش دارد یکی از آنها truncate LSB است که برای این کاربرد دقت کافی را دارد.

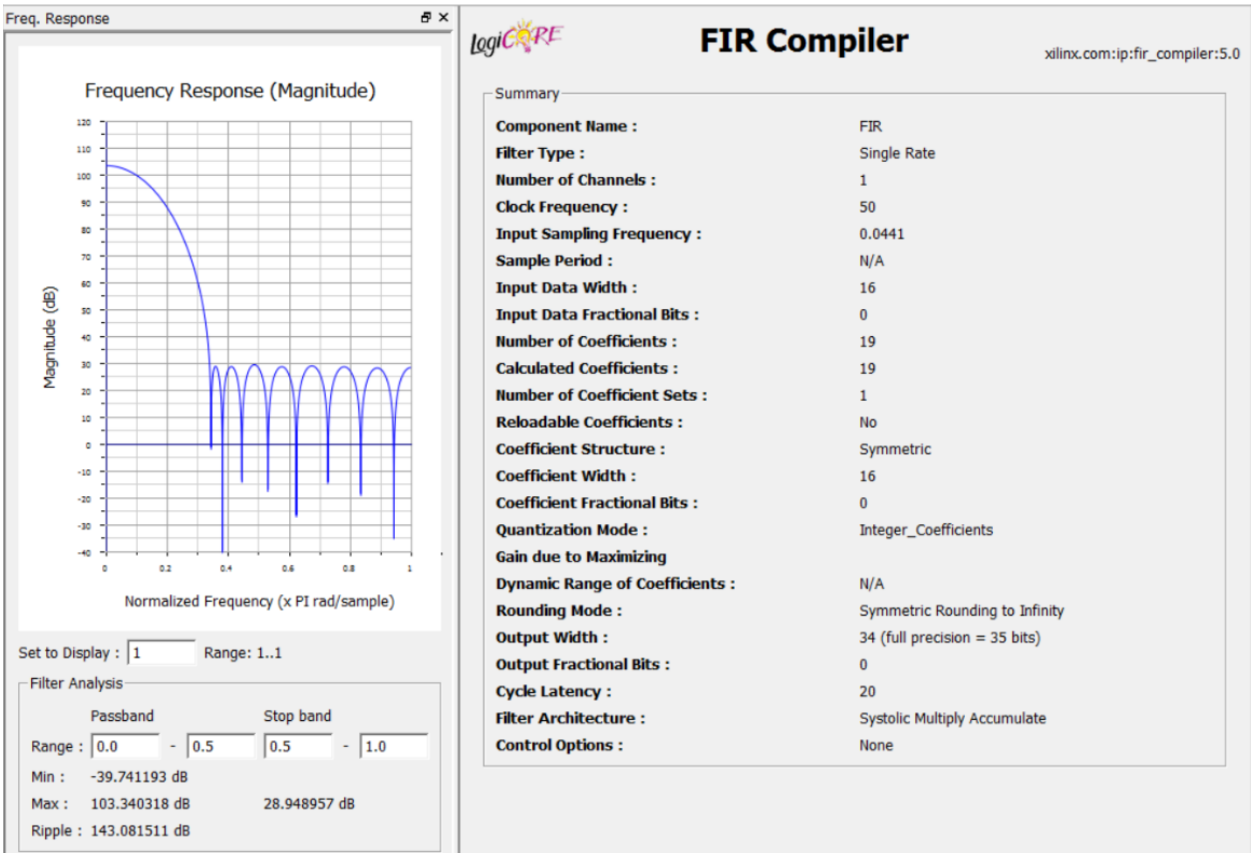
فقط تایم پاسخ دهی اینا خیلی زیاده ☺

میزان فرکانس قابل تنظیمه الان 50MHz

و این فاز به شدت بهتر از فاز قبلی است چون تا 300MHzقابل تنظیم است.و سرعت بسیار بیشتری دارد.

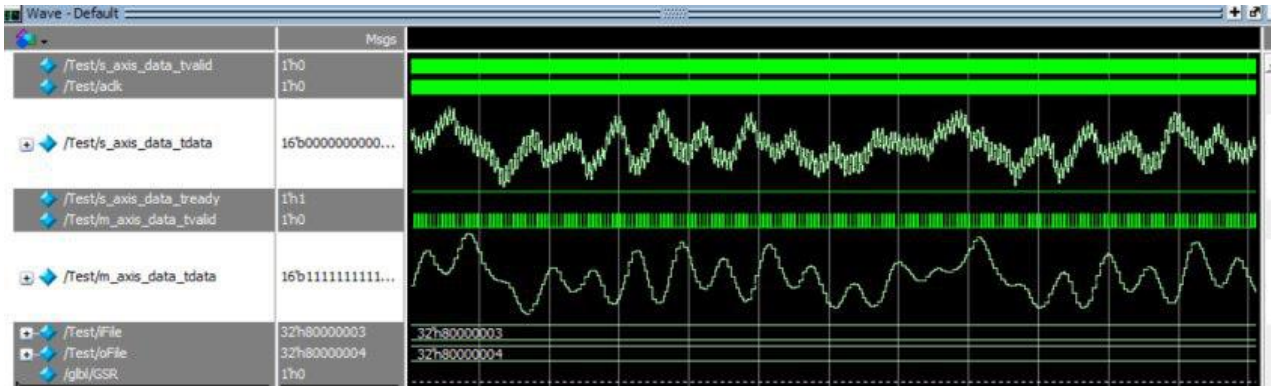


شکل موج فاز V5 3



خلاصه طراحی فاز 3 V5

البته ما توانستیم با V6 هسته هم این قسمت را پیاده سازی کنیم که شکل آن را در پایین مشاهده می فرمایید.



شکل موج فاز 3 V6

Summary

Component Name :	FirCore
Filter Type :	Single Rate
Number of Channels :	1
Clock Frequency :	300
Input Sampling Frequency :	4.41
Sample Period :	N/A
Input Data Width :	16
Input Data Fractional Bits :	0
Number of Coefficients :	19
Calculated Coefficients :	19
Number of Coefficient Sets :	1
Reloadable Coefficients :	No
Coefficient Structure :	Non Symmetric
Coefficient Width :	16
Coefficient Fractional Bits :	0
Quantization Mode :	Integer_Coefficients
Gain due to Maximizing	
Dynamic Range of Coefficients :	N/A
Rounding Mode :	Truncate LSBs
Output Width :	16 (full precision = 34 bits)
Output Fractional Bits :	0
Cycle Latency :	25

خلاصه طراحی فاز 3 V6

4. نتیجه گیری

ما می توانیم بر اساس نیاز خودمون سراغ هر یک برویم ولی باید قبول کنیم :

1. هر چه قدر خوب هم طراحی کنیم باز محدودیت حافظه و کلاک داریم.
2. با استفاده از pipeline میتوانیم بازدهی را بسیار افزایش دهیم.
3. ولی سرانجام با core فهمیدیم قابلیت های کد های ما بسیار نزدیک مهندسان xillinxاست. و با تلاش و تغییر میتوان به آن رسید.