

67707

YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

MİKROİŞLEMCİLİ FİLTRE TASARIMI



Elektronik Müh. Galib DEMİRTAŞ

**F.B.E. Elektronik ve Haberleşme Mühendisliği Anabilim Dalında
hazırlanan**

YÜKSEK LİSANS TEZİ

Tez Danışmanı

:Doç.Dr. Ertuğrul ERİŞ

(Handwritten signature)
(Handwritten signature)
(Handwritten signature)

İSTANBUL , 1997

İÇİNDEKİLER

TEŞEKKÜR	II
ÖZET	III
SUMMARY	V
1. GİRİŞ	1
1.1 Dijital Sinyal İşleme Sistemleri	2
1.2 Analog Sinyalin Örneklenmesi	6
1.3 Örnekleme Teoremi	8
1.4 Kuantalama ve Kodlama	10
1.5 Dijitalden Analoga Çevrim	11
2. DİJİTAL FİLTRE TASARIM TEKNİKLERİ	11
2.1 FIR Filtreler	11
2.2 IIR Filtreler	13
2.2.1 Zaman Bölgesinde Sentez	14
2.2.2 Frekans Bölgesinde Dizayn	19
3. DİJİTAL FİLTRENİN GERÇEKLENMESİ	25
3.1 Dijital Filtre Dizaynı	27
3.2 Doğrudan Gerçeklemeler	27
3.2.1 1D Yapısı	27
3.2.2 2D Yapısı	29
3.2.3 3D Yapısı	31
3.2.4 4D Yapısı	32
3.3 Ardışıl (Cascade) Gerçeklemeler	33
3.4 Paralel Gerçeklemeler	37
4. MİKROKONTROLÖR İLE (80C32) DİJİTAL FİLTRENİN TASARIMI	44
4.1 Aritmetik İşlemlerin Gerçekleştirilmesi	44
4.2 Filtre Katsayılarının Bulunması ve Filtrenin Gerçeklenmesi	45
5. SONUÇ	50
KAYNAKLAR	51

EKLER	52
EK1: 80C32 Filtre Programı	52
EK2: Matlabte Filtre Katsayılarının Bulunması	65
EK3: Pascal İle Filtre Katsayılarının Aktarılması	74
EK3: Protel Devre Şeması	77
EK4: Smart PCB Çizimleri	79
ÖZGEÇMİŞ	81



TEŐEKKÜR

Bu tezi bana öneren ve gerçekleřtirmemde yardımlarını esirgemeyen tez hocam Sn. Doç. Dr. Ertuğrul ERİŐ'e ve hocam Yrd. Doç. Dr. Tuncay UZUN'a teőekkürlerimi sunmayı bir borç bilirim.

Yine tezimde fikir alış verişinde bulunduğum Araő. Gör. Beőir TAYFUR ve Arő. Gör. Ercan ÖZGENCİL'e teőekkürlerimi sunarım..

Yüksek Lisans Tezimi, manevi desteğini esirgemeyen arkadaşım Aysel ŐEN'e ithaf ediyorum.

ÖZET

Mikrokontrollör kullanılarak dijital filtre tasarımı yapılmıştır. Önce istenen filtre karakteristiğini seçilen bir yaklaşıklıkla sağlayan filtrenin s-domenindeki transfer fonksiyonu bilgisayar ortamında elde edilmiştir. Sonra bu transfer fonksiyonun bilinear-z dönüşümüyle bulununan z-domenindeki transfer fonksiyonu, yine bilgisayar ortamında bulunmuştur.

ADC ile bulunan analog işaretin karşılığı dijital işaret, bilgisayar ortamından elde edilen z-domenindeki istenen filtrenin transfer fonksiyonu ile 80C32 mikrokontrollör ortamında işleme tabi tutularak filtre çıkışındaki dijital işaret elde edilmiştir. Bu amaçla 1D (first direct) algoritması kullanılmıştır. Bir sonraki adımda, filtre çıkışındaki dijital işaret DAC ile analog işarete dönüştürülmüştür.

ABSTRACT

Digital filter design is realised by means of microcontrollers. Firstly s-domain transfer function of the required filter with a chosen approximation is determined by computer. Secondly bilinear-z transformed z-domain transfer function corresponding to the required filter is also determined.

In order to find digital filter output, digital signal corresponding to analog signal and found by ADC is processed with the z-domain transfer function of the required filter on a 80C32 microcontroller. First direct algorithm is used for this purpose. In the next step filter output digital signal is transferred to analog signal through a DAC chip.

1. GİRİŞ

VLSI (Very Large Scale Integration Circuits) teknolojisindeki gelişmeler mikrobilgisayarlara, A/D ve D/A çeviricilere ve elektronik elemanlara yeni üstünlükler kazandırmaktadır. Yine bunların kullanıldığı yapılar olan dijital filtrelerde böylece analog filtrelere göre üstün özelliklere sahip olmaktadır. Ayrıca mikroişlemci ve çevre elemanlarının ucuzlaması ile dijital filtre yapımında maliyet azalmakta ve pratik uygulamalarında da artma olmaktadır.

Diğer yandan dijital filtreler için analog filtrelere göre aşağıdaki üstünlükleri sıralayabiliriz.

- 1) Frekans karakteristiğinde kayma yoktur, kararlıdır.
- 2) Düşük frekanslı işaretlerde yüksek doğruluk elde edilir.
- 3) Frekans cevabı karakteristiği, analog filtrelere göre ideale çok yakın yapılabilir.
- 4) Lineer faz karakteristiği elde edilebilir.
- 5) Adaptif filtre yapmak mümkündür.
- 6) Seçilen çeviricilere (ADC ve DAC) bağlı olarak örnekleme hassasiyeti ve filtre doğruluğu kesinlikle kontrol edilebilir.
- 7) Donanım maliyeti düşük ve sistemlerle uyumsuzluk problemi yoktur.

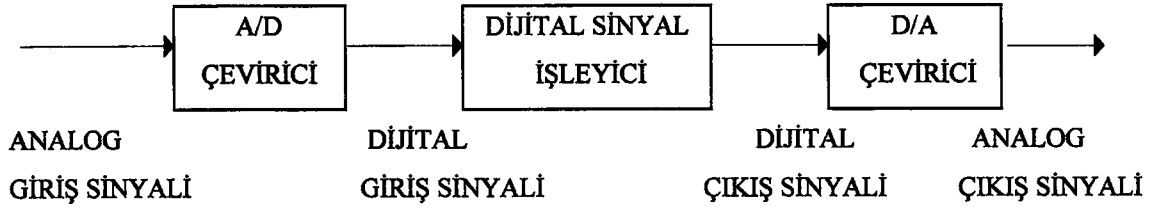
Dijital bir filtrede $x(t)$ analog giriş işareti T anlarında örneklenip $x^*(t)$ örneklenmiş işareti elde edilir. Her örnek dijital bir bilgiye çevrilip işlemcide işlenmekte (filtre programında) ve $y^*(t)$ dijital çıkış işareti elde edilmektedir. Bu bilgi $y(t)$ analog sinyaline çevrilip filtre edilmiş yeni kontrol sinyali elde edilir.

1.1 DİJİTAL SİNYAL İŞLEME SİSTEMLERİ

Sinyal, zamanla, konumla veya herhangi bir bağımsız değişken veya değişkenlere bağlı değişim gösteren fiziksel büyüklük olarak tanımlanabilir. Sistem, sinyal üzerinde belirlenmiş bir operasyonu gerçekleştiren fiziksel cihaz olarak tanımlanabilir.

Bilimde ve mühendislikte çoğu sinyal analogtur. Yani sinyaller sürekli değişkenlerin fonksiyonlarıdır; zaman, yer gibi; genelde sürekli geniş aralıklarda değişirler. Bu sinyaller filtreler (analog), frekans analizörleri, frekans çoğullayıcı gibi karakteristikleri değişen amaçlar için direkt olarak devrelere uygulanabilirler. Buna benzer durumda sinyal doğrudan analog formda işlenmektedir.

Dijital sinyal işleme, analog sinyal işlemeye alternatif bir metod sağlar. Dijital sinyal işlemenin uygulanmasında, analog sinyal ve dijital işlemci için ara devre gerekmektedir. Bu ara devre Analog-Dijital Çevirici (ADC) olarak adlandırılır. A/D çeviricinin çıkışı dijital işlemcinin girişlerine uygun haldeki dijital sinyaldir. Dijital sinyal işleyici, giriş sinyali üzerinde istenen operasyonları yapan programlanabilir dijital bilgisayar veya mikroişlemci olabilir. Hatta belirli işlemleri gerçekleştirecek şekilde tasarlanmış mekanik (hardwired, donanımsal) dijital işleyici olabilir. Programlanabilir makineler, mekanik (hardwired) makinenin yeniden düzenlenebilirliğinin zorluğundan dolayı yazılımlarında sinyal işleme operasyonlarını değiştirebilecek esnekliğe sahip olmalılar. Genelde programlanabilir sinyal işleyiciler pratikte daha yaygındırlar. Diğer yandan sinyal işleme operasyonu, bazı uygulamalar için iyi tanımlandığı zaman, hardwired (donanımsal) sinyal işleme operasyonları optimize edilebilir ve bu konfigürasyon daha ucuza gelebilmekte ve genelde hızlıda olabilmektedir. Dijital sinyal işleyicinin dijital çıkışlarının kullanıldığı uygulamalarda, çıkışların kullanıcıya analog formda verilmesi gerekmektedir. Buda dijitalden analog domaine başka bir ara yüzeyi gerektirir. Bu devreler Dijital-Analog Çevirici olarak adlandırılır.



ŞEKİL 1.1 Analog Sinyalin İşlenmesi

Analog sinyalin direkt olarak analog bölgede işlenmeyip, dijital sinyal işleyicide işlenmesinin nedenleri:

a) Dijital programlanabilir sistemlerin, programdaki basit değişikliklerle dijital sinyal işleme operasyonunu yeniden düzenleyebilme esnekliğine sahip olabilmesi. Analog sistemin yeniden düzenlenmesi demek sistemin yeniden dizaynı, testi ve kurulması demektir.

b) Hassiyet ölçütleri ayrıca önemli rol oynar. Analog devredeki elemanların toleranslarının kontrolünün zor olması, tasarımcı için analog sinyal işleyici sistemin hassasiyet kontrolünü güçleştirir. Diğer yandan dijital sistemler hassasiyet sorununu en iyi şekilde giderir. Benzer ihtiyaçlar sonuçta A/D çeviricinin hassasiyetinin belirlenmesini ve dijital işlemcinin adres ve veri yolu yapısını, aritmetik işlemcisini ve benzer faktörleri etkiler.

c) Dijital sinyaller kolayca manyetik ortamlara kayıp sorunu olmadan kaydedilebilirler.

A/D ve D/A ÇEVİRİM

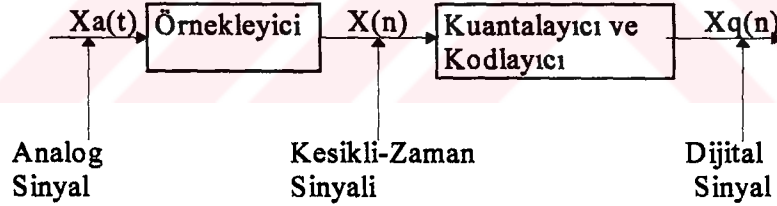
İlgilediğimiz çoğu sinyal; ses, biyolojik sinyaller, sismik sinyaller, radar sinyalleri, sonar sinyalleri ve değişik haberleşme sinyalleri (konuşma ve görüntü); analogtur. Analog sinyali dijital sistemde işlemek demek, birinci olarak belli hassasiyete sahip sayılar dizisine çevirip dijital forma dönüştürmeyi gerektirir. Bu çevirim Analogtan Dijitale çevirim (A/D

çevirim) olarak adlandırılır ve bu işlemi gerçekleştiren cihazlara Analog Dijital çeviriciler (Analog to Digital Convertor) denir.

Kavram olarak analogtan dijitale çevrim iki basamakta gerçekleştirilir.

1) **ÖRNEKLEME:** Bu, sürekli zaman-sinyalinin kesikli-zaman sinyaline dönüştürülmesidir. $X_a(t)$ örneklenecek giriş sinyali ise, çıkış $X_a(nT)=X(n)$ dir. (T örnekleme süresi aralığıdır.)

2) **KUANTALAMA ve KODLAMA:** Bu kesikli-zaman sürekli-değerli sinyalin kesikli-zaman kesikli-değerli sinyale çevrilmesidir. Her örneklenmiş sinyalin değeri sonlu değerli mümkün olabilen değerlerle gösterilir. Kodlama işleminde, her kesikli değer b bit sayıyla; $X_q(n)$; gösterilebilir. Kuantalanmamış örnekle; $X(n)$; kuantalanmış örnek; $X_q(n)$; arasındaki fark kuantalama hatasıdır.



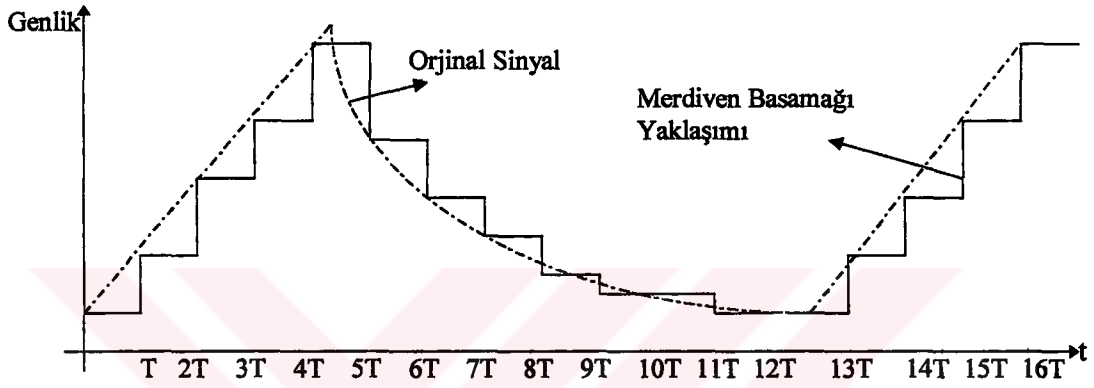
Şekil 1.2 Analog-Dijital Çeviricinin Modellenmesi

A/D çeviriciyi, bir örnekleyci ve onun çıkışıda kuantalayıcı-kodlayıcıya bağlı olan bir yapı olarak modelliyebiliriz. Gerçekte A/D çevirimi, girişi $X_a(t)$ olan ve çıkışta $X_q(n)$ üreten tek bir cihaz tarafından gerçekleştirilir.

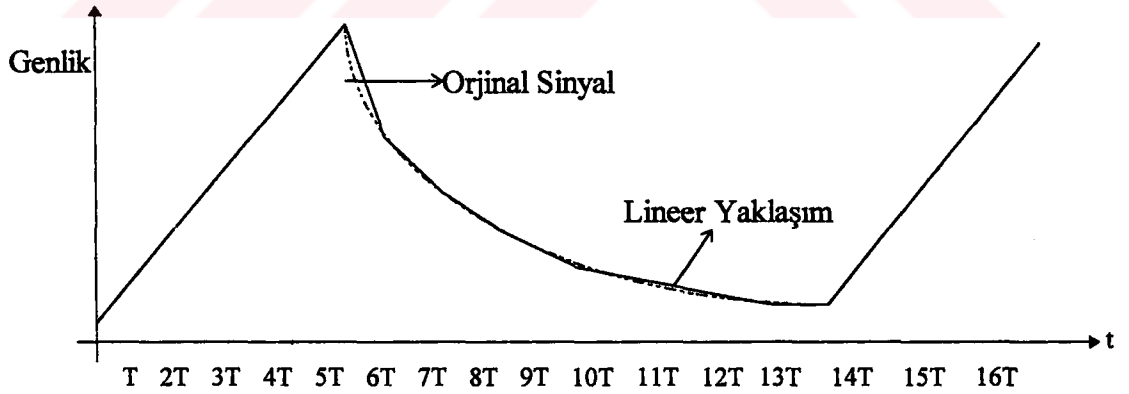
Bazı durumlarda ilgilenilen, işlemden geçen dijital sinyalin analog formda olmasıdır. Dijital sinyalin analog sinyale çevrim işlemi Dijitalden Analoga çevrim olarak (DAC) adlandırılır. Tüm D/A çeviriciler D/A çevirme işlemini, hassiyetine bağlı olarak

çeşitli interpolasyon teknikleri kullanarak dijital sinyaldeki noktaların birleştirilmesi şeklinde gerçekleştirirler.

Şekil 1.3, Dijital -Analog çevrimin merdiven basamağı (stair case approximation) olarak adlandırılan basit bir formunu göstermektedir. Şekil 1.4 ise lineer yaklaşım metodunu göstermektedir.



Şekil 1.3 Sıfırıncı dereceden tutucu Dijital-Analog Çevrimi



Şekil 1.4 Birinci dereceden tutucu Dijital-Analog Çevrimi

Temelde analog sinyal, örnekleme oranı yanlış örneklemeden (aliasing) kaçınacak şekilde yeterince yüksek tutulursa alınan örneklerden geri elde edilebilir. Diğer yandan, kuantalama, sinyal bozukluğu ve gürültülerden dolayı eski haline döndürülemez bir

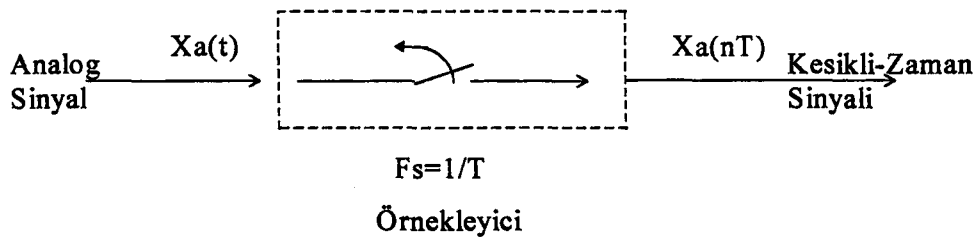
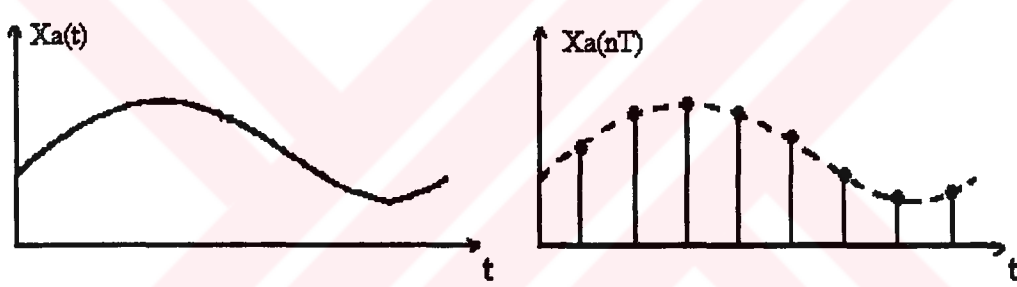
işlemdir. Bozukluğun (signal distortion) büyüklüğü A/D çevirme işleminin hassiyetine (ölçümün kaç bitlik olduğuna) bağlıdır. Fiyat ve örnekleme oranı A/D çevirici için istenen hassiyeti etkiler. Genelde fiyat hassaslık ve/veya örnekleme oranındaki artışla artar.

1.2 ANALOG SİNYALİN ÖRNEKLENMESİ

Analog sinyalin örneklenmesi için bir çok yol vardır. Burada pratikte en çok kullanılan örnekleme tipleri olan periyodik ve üniform örnekleme ile ilgileneceğiz.

$$x(n)=x_a(nT) \quad -\infty < n < \infty \quad (1.1)$$

$x(n)$, analog sinyalden; $x_a(t)$; her t saniyede alınan örneklerle elde edilmiş kesikli-zaman sinyalidir. Bu işlem Şekil 1.5' de gösterilmiştir.



Şekil 1.5 Analog sinyalin periyodik örneklenmesi

T, zaman aralığı, örnekleme periyodu olarak adlandırılan iki örnek arasındaki süredir. $1/T=F_s$ ise örnekleme frekansdır. Periyodik örnekleme, sürekli-zaman ve kesikli-zaman sinyallerinin zaman değişkenleri arasında bir bağlantı kurar. Gerçekten bu değişkenler T örnekleme periyodu ile lineer bağımlıdır.

$$t=nT=n/F_s \quad (1.2)$$

Analog sinyalin frekans değişkeni F (veya Ω) ile kesikli-zaman sinyalinin frekans değişkeni f (veya ω) arasında bir bağlantı vardır.

Analog sinüzoidal sinyal ;

$$X_a(t) = A \cos(2\pi Ft + \theta) \quad (1.3)$$

şeklindedir. $F_s=1/T$ frekansında örneklendiğinde örneklenmiş sinyal, kesikli-zaman sinyali;

$$\begin{aligned} X_a(nT) &= X(n) = A \cos(2\pi Fnt + \theta) \\ &= A \cos(2\pi nF / F_s + \theta) \end{aligned} \quad (1.4)$$

F ve F_s birbirleriyle lineer bağımlıdır.

$$f = F / F_s \quad (1.5)$$

$$\omega = \Omega T \quad (1.6)$$

1.5 denkleminde f, normalize frekanstır. F veya Ω , sürekli-zaman sinüzoidal sinyali için;

$$-\infty < F < \infty , \quad -\infty < \Omega < \infty \quad (1.7)$$

şeklindedir. Buna karşın kesikli-zaman sinüsoidal için f veya ω ;

$$1/2 < f < 1/2, \quad -\pi < \omega < \pi \quad (1.8)$$

biçimindedir. 1.5 ve 1.8 denklemleri yerine konulursa ;

$$F_s/2 \leq F \leq F_s/2, \quad -1/2T \leq \Omega \leq 1/2T \quad (1.9)$$

veya eş olarak;

$$\pi F_s \leq \Omega \leq \pi F_s, \quad -\pi/T \leq \Omega \leq \pi/T \quad (1.10)$$

olur. Bağıntılardan anlaşılacağı üzere temel fark, F ve f (veya Ω ve ω) frekans değişkenlerinin değer aralığındadır.

Sürekli-zaman sinyalin periyodik örnekleme demek F için sonsuz olan frekans aralığının f (ve ω) için sonlu bir aralığa transfer edilmesidir (frequency mapping). Kesikli-zaman sinyalinde en yüksek frekans $\omega=\pi$ veya $f=1/2$ ve örnekleme oranı F_s iken, buna karşılık gelen en yüksek F ve Ω değeri,

$$F_{\max} = F_s/2 = 1/2T, \quad \Omega_{\max} = \pi F_s = \pi/T \quad (1.11)$$

1.3 ÖRNEKLEME TEOREMİ

Verilen bir analog sinyal için örnekleme periyodu T (F_s)'yi nasıl seçmeliyiz. Bunun için örneklenecek sinyalin karakteristiği hakkında biraz bilgi sahibi olmamız gereklidir. Benzer bilgiler genellikle hazırdır. Özellikle sinyalin frekans bileşenleri hakkında biraz bilgi sahibi olmamız gerektirir. Benzer bilgiler genellikle hazırdır.

Mesela, ses sinyali için temel frekans bileşeninin 3KHz civarında olduğunu biliriz. Diğer yanda TV sinyalleri genelde 5MHz'de önemli frekans bileşenlerini içerir. Benzer sinyaller için bilgiler genlikte, frekans ve değişen frekans bileşenlerinin fazlarında vardır. Fakat temelde bu sinyallerin karakteristiği hakkında bilgi mevcut değildir.

Farzedelim bir analog sinyal farklı genliğe, frekansa ve faza sahip sinüsoidallerin toplamı şeklinde ifade edilsin.

$$X_a(t) = \sum_{i=1}^N A_i \cos(2\pi F_i t + \theta_i) \quad (1.12)$$

N, frekans bileşenlerinin sayısıdır. Genlik, frekans ve faz bir zaman aralığından diğerine genellikle yavaş değişirler. Farzedelim frekanslar belli bir frekansı, F_{\max} 'ı aşmıyorlar. Bu bilinen F_{\max} 'tan uygun örnekleme frekansı seçebiliriz. Sinyal " $F_s=1/T$ " frekansında örneklendiği zaman, sinyalin yeniden hatasız geri elde edilebilmesi için analog sinyalin max frekansının " $F_s/2$ " olması gerektiğini biliyoruz. Yanlış örnekleme (aliasing) kaynaklanan bozukluklardan kaçınmak için örnekleme frekansı yeterince büyük seçilmelidir. Yani

$$F_s / 2 > F_{\max} , \quad F_s > 2F_{\max} \quad (1.13)$$

olmalıdır. F_{\max} analog sinyalin frekans bileşenlerinin en büyüğüdür. Seçilen örnekleme frekansı ile analog sinyaldeki herhangi frekans bileşeni, $|F_i| < |F_{\max}|$, kesikli-zaman sinüs sinyaline,

$$-1/2 \leq f_i = F_i / F_s \leq 1/2 \quad (1.14)$$

frekansı ile haritalanır (mapping). Eş olarak,

$$-\pi \leq \omega_i = 2\pi f_i \leq \pi \quad (1.15)$$

$|f| = 1/2$ (veya $|\omega| = \pi$) kesikli zaman sinyalde en yüksek frekans iken yanlış örnekleme probleminde kaçarak 1.13'e göre örnekleme frekansı seçilir. Yani analog sinyaldeki tüm sinüsoidal bileşenler, $F_s > 2F_{\max}$ koşulu altındaki örneklemede, temel (fundemantal) aralıktaki frekanslarla birlikte kesikli-zaman frekans bileşeni karşılıklarına transfer edilirler. Analog sinyalin tüm frekans bileşenleri belirsizlik içermeden örneklenmiş formda ifade edilebilirler; yani analog sinyal uygun interpolasyon metodları kullanılarak (DAC) örneklenmiş değerlerden herhangi bir bozulma olmadan geri elde edilebilir.

ÖRNEKLEME TEOREMİ : Analog sinyal $X_a(t)$ 'nin içerdiği en yüksek frekans $F_{\max} = B$ ise, örneklenen değerlerden sinyali tam olarak geri elde edebilmek için örnekleme frekansı $F_s > 2F_{\max}$ seçilmelidir.

1.4 KUANTALAMA VE KODLAMA

Görüldüğü gibi, dijital sinyal sonlu hassasiyete sahip sonlu sayıda dijitlerden oluşan sayılar dizisidir. Her bir örneklenmiş değer sonlu sayıda dijitle ifade edilmesiyle kesikli-zaman sinyalin dijital sinyale çevrilmesine kuantalama denir. Sürekli-değerli sinyalin sonlu-değerli kesikli-değer seviyeleriyle ifadesinde görülen hataya, kuantalama hatası veya kuantalama gürültüsü denir. $X(n)$ örnekleri üzerindeki kuantalama operasyonu $Q[X(n)]$ olarak gösterilirse; $X_q(n)$ kuantalayıcının çıkışındaki kuantalanmış örneğin dizisini gösterir. Kuantalama hatası, kuantalanmış değerle gerçek değer arasındaki farktır.

$$e_q(n) = X_q(n) - X(n) \quad (1.16)$$

1.5 DİJİTALDEN ANALOĞA ÇEVİRİM

Dijital sinyali analog sinyale çevirmek için Dijitalden Anloğa Çeviriciler (D/A) kullanılır. Bir Dijital-Analog Çevirici, kullandığı bit sayısı ve kullanıcı tarafından belirlenen örnekleme periyodu ile karakterize edilebilir. D/A çeviricinin faaliyeti örnekler arasında interpolasyon yapmasıdır.

Örnekleme teoremi band-sınırlı bir sinyal için optimum interpolasyonu belirler. Fakat, bu tip interpolasyon çok karmaşık ve pratik değildir. Pratikte, en basit D/A çevirici sıfıncı dereceden tutucudur. Lineer interpolasyon kullanarak, Şekil 1.4'deki gibi örnekleri doğru parçalarıyla birleştirerek, geliştirmeler yapılabilir.

2. DİJİTAL FİLTRE TASARIM TEKNİKLERİ

Lineer sabit katsayılı filtreler iki genel sınıfa ayrılırlar; sonlu-tepki cevabı (FIR, Finite Impulse Response) filtreler ve sonsuz-tepki cevabı (IIR, Infinite Impulse Response) filtreler. IIR filtreler geri besleme içeren yapılarda, yani tekrarlı (recursive structures) yapıların gerçekleştirilmesinde kullanılırlar. FIR filtreler ise geri besleme içermeyen (tekrarsız, (nonrecursive)) yapılarda kullanılırlar.

2.1 FIR FİLTRELER

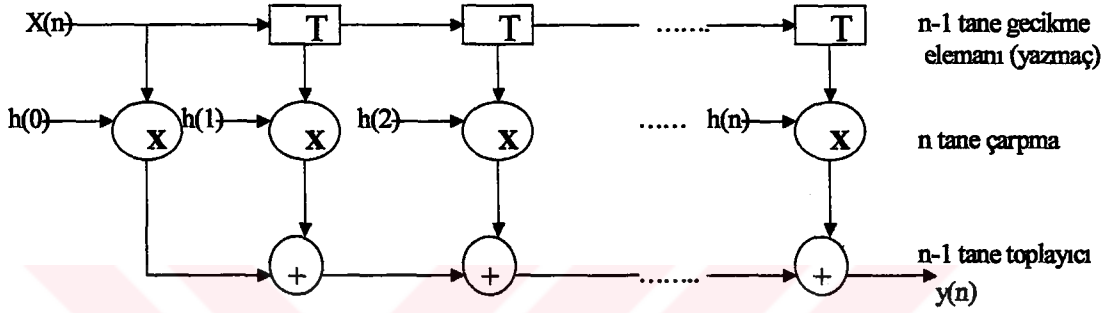
Bir FIR filtre fark denklemi cinsinden aşağıdaki gibi ifade edilir.

$$y(n) = c_0 \cdot x(n) + c_1 \cdot x(n-1) + \dots + c_{N-1} \cdot x(n-N+1) \quad (2.1)$$

Yukarıdaki fark denklemi transfer fonksiyonu olarak ise 2.2'deki gibi gösterilebilir.

$$H(z) = \sum_{i=0}^{N-1} c_i \cdot z^{-i} \quad (2.2)$$

Tipik bir n. dereceden FIR filtre Şekil 2.1'de gösterilmiştir.



Şekil 2.1 FIR Filtre Mimarisi

FIR filtre yapısı incelendiğinde çeşitli özellikler sergilediği görülür. Bunlar;

1. Filtrenin tepki cevabı n.ci örnekten sonra görülür.
2. Filtre transfer fonksiyonu sadece sıfırları içermektedir.
3. Filtre basit bir dizayna sahiptir. Sadece yazmaçlar, toplayıcılar ve çarpıcılar içermektedir.
4. Eğer girişler sınırlı ise ($|x(i)| \leq 1$) ise çıkışın max. değeri $y(i) \leq \sum |c_i|$ olacaktır.
5. Frekansa göre fazı incelendiğinde sabit eğimli lineer bir doğru olduğu görülür.

2.2 IIR FİLTRELER

Önceki bölümde görüldüğü gibi FIR filtreler süperbilineer faz davranışı gösterirler. Fakat iyi kalitede (dik eğimli) genlik-frekans cevabı başarmak isteniyorsa, bu yüksek dereceden FIR filtre gerektirmektedir. FIR ve IIR filtreler karşılaştırıldığında;

1) Verilen genlik-frekans cevabını sağlayan filtrenin IIR filtrelerle daha düşük dereceden gerçekleştirildiği;

2) Genellikle lineer faz veya sabit grup gecikmesi davranışını sergilemez.

Digital filtre tasarımında temel nokta belirlenmiş olan genlik frekans cevabının sağlanması ise IIR filtreler bunu daha düşük dereceden (FIR filrelere göre) bir yapıyla gerçekleştirirler. Buda daha az sayıda çarpma ve hafızaya kayıt yapılması ve hafızada yer ayrılması demektir. Yani işlem hızının artması, dolayısıyla örnekleme hızının artması demektir.

IIR filtrenin tepki cevabı (impulse response) çok uzun veri dizileridir. Filtre transfer fonksiyonu aşağıdaki gibi gösterilebilir.

$$H(z) = \frac{N(z)}{D(z)} = \sum_{n=0}^{\infty} h(n).z^{-n} = \frac{\sum_{i=0}^M b_i . z^{-i}}{1 + \sum_{i=1}^N a_i . z^{-i}} = K \frac{\prod_{i=1}^M (z - \alpha_i)}{\prod_{i=1}^N (z - \beta_i)} \quad (2.3)$$

veya açık olarak;

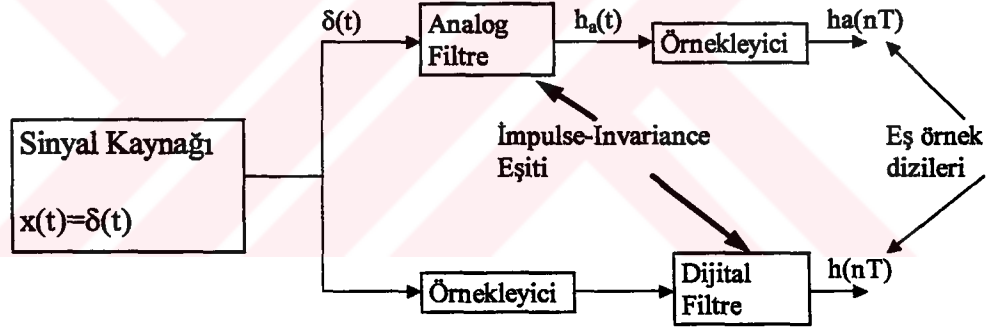
$$H(z) = \frac{b_0 + b_1 . z^{-1} + b_2 . z^{-2} + \dots + b_n . z^{-M}}{1 + a_1 . z^{-1} + a_2 . z^{-2} + \dots + a_n . z^{-N}} \quad (2.4)$$

Burada temel problem, filtrenin lineer fark denkleminin katsayılarını oluşturan a_i ve b_i değerlerinin belirlenen işlevi gerçekleştirecek şekilde hesaplanmasıdır. IIR filtre dizaynında genel başlangıç noktası analog (sürekli) transfer fonksiyonudur, $H_a(s)$. Buradanda kesikli-zamandaki sistem fonksiyonuna geçilir. Dijital filtre fonksiyonunun analog prototipinden elde edilmesi işlemi, zaman ve frekans bölgelerinde olmak üzere iki ayrı yapıda elde edilebilir.

2.2.1 ZAMAN BÖLGESİNDE SENTEZ: SABİT (INVARIANT) TASARIM

- **Impulse-Invariant Tasarım (Tepkisi Değişmeyen Sistemler)**

Bir impulse-invariant dijital filtre için sentez tekniği Şekil 2.2’de gösterilmiştir.



Şekil 2.2 İmpulse İnvariance Tekniği

Analog filtre çıkışı, $h_a(t)$, birim-impulse cevabıdır (unit-impulse response). Bu impulse cevabının örneklenmesi, $h_a(nT)$, yine benzer çıkış değerlerini verir. Birim-impulse fonksiyonunun kesikli-zaman karşılığı birim-pulse'tır. Birim-pulse'ların büyüklüğü örneklenen değer genliğine bağlıdır. Dijital filtrenin girişleri birim-pulse ve çıkışları birim-pulse cevabı olacaktır. Eğer dijital filtrenin parametreleri, birim-impulse tepkisi önceden belli olan $h_a(nT)$ değerleriyle aynı olacak şekilde ayalanırsa; dijital filtreye analog filtrenin *impulse-invariant* eşdeğeri denir.

İmpulse-invariant sentez tekniğini, transfer fonksiyonu m farklı gerçek kutup içeren bir analog filtre transfer fonksiyonu üzerinde görelim. Filtre transfer fonksiyonunun kısmi parçalara ayrılmış (partial fraction expansion) ifadesi $H_a(s)$;

$$H_a(s) = \sum_{i=1}^m \frac{K_i}{s + s_i} \quad (2.5)$$

s_i 'ler filtrenin kutupları ve K_i, s_i kutbu için fonksiyonun genlik değeri. Birim-impulse girişi için ters-Laplace transformu alınır;

$$h_a(t) = \sum_{i=1}^m K_i \cdot e^{-s_i t}, t \geq 0 \quad (2.6)$$

olur. Birim-impulse cevabın dönüşümü alınır;

$$H_1(z) = \sum_{n=0}^{\infty} h_a(nT) \cdot z^{-n} \quad (2.7)$$

ve 2.6 ifadesi yerine konulursa;

$$H_1(z) = \sum_{n=0}^{\infty} \sum_{i=1}^m K_i (e^{-s_i T} \cdot z^{-1})^n \quad (2.8)$$

şeklinde sonucu elde ederiz. İfadeyi düzenlersek;

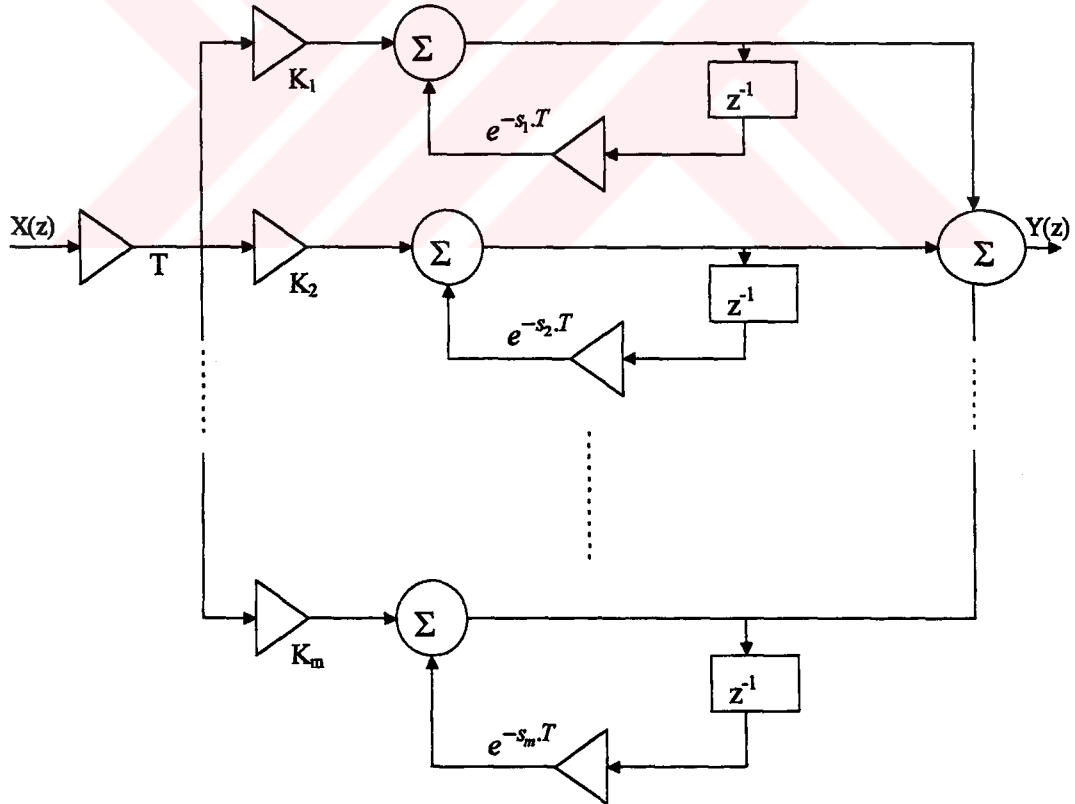
$$H_1(z) = \sum_{i=1}^m K_i \cdot \sum_{n=0}^{\infty} (e^{-s_i T} \cdot z^{-1})^n \quad (2.9)$$

$$H_1(z) = \sum_{i=1}^m \frac{K_i}{1 - e^{-s_i T} \cdot z^{-1}} \quad (2.10)$$

olarak transfer fonksiyonunun z-dönüşümünü elde ederiz. 2.10 ile tanımlanan filtre, analog filtrenin örneklenmiş impulse tepkisi ile eş olan birim pulse tepkisine sahip olacaktır. Daha sonra göreceğimiz gibi dijital filtrenin genlik cevabı örneklemeden dolayı f_s örnekleme frekansı ile ölçeklenmelidir. Dijital filtrenin genliğinin ölçeklenmesi analog filtrenin genlik cevabına yaklaşık olarak eşitlemek için yapılır. Bu $H_1(z)$ 'nin $T=1/f_s$ ile çarpılması demektir. İnvaryant-dijital filtrenin sonuç pulse transfer fonksiyonu 2.11'deki gibi olur.

$$H(z) = T \cdot \sum_{i=1}^m \frac{K_i}{1 - e^{-s_i T} \cdot z^{-1}} \quad (2.11)$$

Yukarda anlatılan sentezleme tekniğinin (İmpulse-invariant sentez) paralel gerçekleştirilmesi Şekil 2.3 ile gösterilmiştir.



Şekil 2.3 İmpulse İnvaryant Filtre Paralel Gerçeklemesi

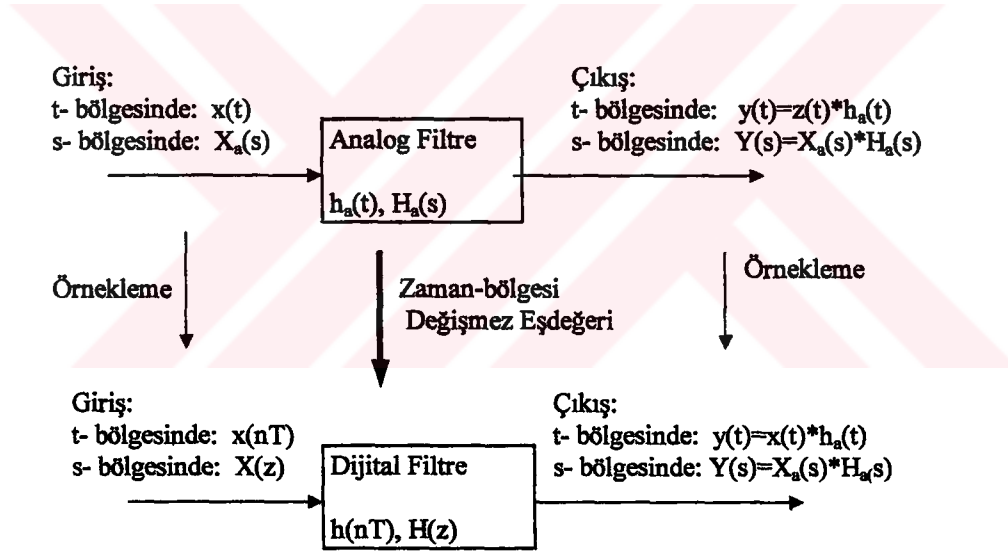
Analog filtrenin $s_i = \sigma_i + j\omega_i$ kutupları dönüşüm sonucunda $H(z)$ ' nin;

$$z_i = e^{s_i T} = e^{(\sigma_i + j\omega_i)T} \quad (2.12)$$

kutuplarına dönüşür. Buradan, $\sigma_i < 0$ için $|z| < 1$ yani, sol yarı düzlemdeki analog kutuplar z -düzleminde birim daire içine düşmektedir. Kararlı analog filtre, bu yöntemle kararlı dijital filtreye dönüşmektedir.

- Genel Zamanda-Değişmeyen Sentez (Time-İnvariant Sentez)

Zaman bölgesindeki değişmezlik kavramı Şekil 2.4 ile örneklenmiştir.



Şekil 2.4 Zaman-Bölgesinde Değişmezlik

Dijital filtre girişi $x(nT)$, analog filtre girişi $x(t)$ ' nin örneklenmiş versiyonudur. Analog ve dijital filtreye bu eşdeğer girişler uygulanır ve $H(z)$ ' yi belirleyen filtre katsayıları, analog filtrenin örneklenmiş çıkışlarıyla dijital filtrenin çıkışları aynı oluncaya dek değiştirilir.

İstenen $H(z)$ fonksiyonu aşağıdaki gibi elde edilir.

$$y(t) = L^{-1}[H_a(s).X_a(s)] \quad (2.13)$$

Dijital filtrenin çıkış değerleri aşağıdaki şekilde ifade edilebilir.

$$y(nT) = \left[L^{-1}[H_a(s).X_a(s)] \right]_{t=nT} \quad (2.14)$$

(2.14) ifadesinin z-dönüşümü dijital filtrenin z-bölgesindeki çıkışını verir. Bu bağıntı (2.15) ile verilmiştir.

$$Y(z) = H(z).X(z) = G.L\left\{ \left[L^{-1}[H_a(s).X_a(s)] \right]_{t=nT} \right\} \quad (2.15)$$

L, laplas dönüşümünü göstermektedir. Sabit G değeri benzer frekans tepkisine sahip analog ve dijital filtre fonksiyonları tarafından içerilir. $H(z)$ 2.15 denkleminde çekilirse;

$$H(z) = \frac{G}{X(z)}.L\left\{ \left[L^{-1}[H_a(s).X_a(s)] \right]_{t=nT} \right\} \quad (2.16)$$

İmpulse-İnvariant filtreler belkide zaman bölgesi invariant sentez tekniğini kullanan en popüler dijital filtre sentez tekniğidir. İmpulse invariant girişler için;

$$X(z) = X_a(s) = 1 \quad (2.17)$$

ve

$$G = T \quad (2.18)$$

özellikleri vardır.

2.2.2 FREKANS BÖLGESİNDE DİZAYN: BİLİNEER z-DÖNÜŞÜMÜ

Önceki bölümde anlatılan zaman bölgesinde değişmezlik metodu filtre sentezinde kullanılabilir. Fakat bir çok uygulamalarda aliasing engellemeleri (hatalı örnekleme) sorun yaratır. Bu problemin üstesinden gelen ve $H_a(s)$ 'in ters laplasının alınmasını gerektirmeyen Bilineer z-dönüşümü, zaman bölgesi değişmezlik metodundan daha kolay bir sentez tekniğidir.

Aliasing etkilerinden (hatalı örnekleme) kaçınmak için analog filtrenin transfer fonksiyonu aşağıdaki şekilde band sınırlı olmalıdır.

$$-\frac{1}{2}f_s \leq f \leq \frac{1}{2}f_s \quad (2.19)$$

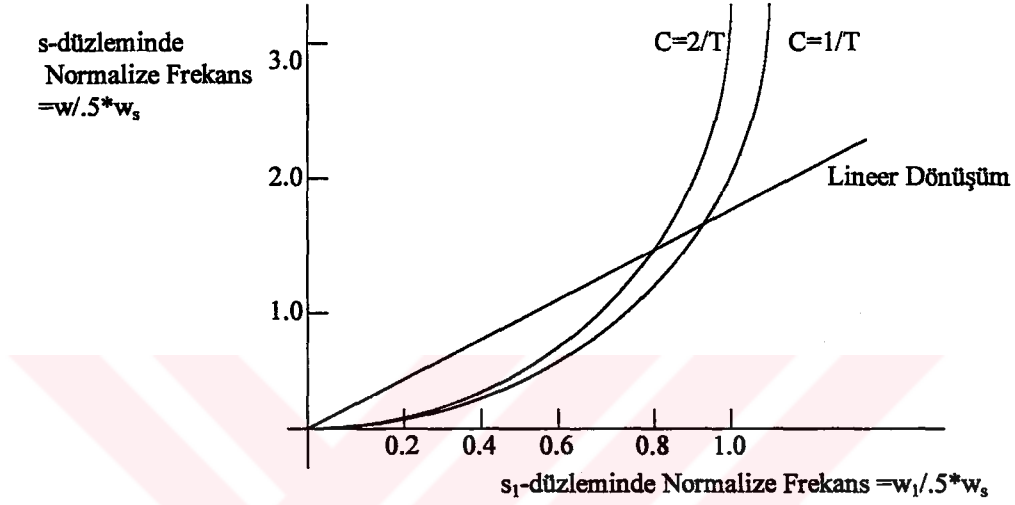
Analog transfer fonksiyonu bu şartı sağlamıyorsa nonlinear dönüşüm yapılarak bandsınırlı olacak şekilde dönüştürülmelidir. Bu teknikle s-düzlemindeki $j\omega$ eksenini s_1 -düzleminde (2.19) aralığına transfer edilecek şekilde tüm s-düzlemi s_1 düzlemine transfer edilir. Bu işlemi gerçekleştiren bir çok dönüşüm vardır. Bunlardan birini inceliyeceğiz. Bu şartları sağlayan dönüşüm;

$$w = C. \tan \left[\frac{w_1 \pi}{\frac{1}{2}w_s} \right] = C. \tan \frac{w_1 \cdot T}{2} \quad (2.20)$$

veya

$$C = w \cdot \cot \frac{w_1 \cdot T}{2} \quad (2.21)$$

Bu dönüşüm Şekil 2.5' te gösterilmiştir.



Şekil 2.5 Bilineer z-dönüşümünde Frekans Aktarılması

C sabiti, istenen her frekans için $w=w_1$ benzerliğini kuracak şekilde seçilir. Örneğin $w=w_1=w_r$ için C çözümlürse;

$$C = w_r \cdot \cot \frac{w_r \cdot T}{2} \quad (2.22)$$

Küçük r değerleri için;

$$\frac{w_r \cdot T}{2} \ll 1 \quad (2.23)$$

varsayabiliriz. Bu durumda küçük x'ler için $\cot x \cong 1/x$ özelliğinden yararlanarak;

$$C = w_r \left(\frac{2}{w_r \cdot T} \right) = \frac{2}{T} \quad (2.24)$$

şeklinde yazabiliriz. Bu bağıntının gerçekleştiğini Şekil 2.5' tede görebiliriz. Diğer yanda $C=1/T$ için ;

$$w = \frac{1}{T} \tan \frac{w_1 \cdot T}{2} \quad (2.25)$$

olarak yazılabilir ve Şekil 2.5' tede görülebileceği gibi $\frac{1}{2} \cdot w_s$ için yapılan normalizde 0.74 normalize frekansta $w=w_1$ olmaktadır. $f=0$ için w ve w_1 daima eşittirler. Yani bilineer z-dönüşümü ile elde edilen dijital filtrenin dc tepkisi ile analog filtrenin dc tepkisi aynı olacaktır.

s ile s_1 arasındaki bağıntı (2.20) denkleminde kolayca bulunabilir.

$$\tan x = \frac{\sin x}{\cos x} = -j \cdot \frac{e^{jx} - e^{-jx}}{e^{jx} + e^{-jx}} \quad (2.26)$$

olduğunu hatırlarsak ve $s=jw$ ve $s=jw_1$ için (2.20) denklemini düzenlersek ;

$$s = C \cdot \frac{e^{s_1 \cdot T/2} - e^{-s_1 \cdot T/2}}{e^{s_1 \cdot T/2} + e^{-s_1 \cdot T/2}} = C \cdot \tanh \frac{s_1 \cdot T}{2} \quad (2.27)$$

şeklinde olur. Dijital filtre, $H_a(s_1)$ fonksiyonunda ;

$$z = e^{s_1 \cdot T} \quad (2.28)$$

yazılarak bulunabilir. Bu değişiklikle s aşağıdaki gibi yazılır.

$$s = C. \frac{1 - e^{-s_1 T}}{1 + e^{-s_1 T}} = C. \frac{1 - z^{-1}}{1 + z^{-1}} \quad (2.29)$$

Asıl hedefimiz olan $H(z)$ dijital filtre transfer fonksiyonu $H_a(s)$ transfer fonksiyonunda s yerine ;

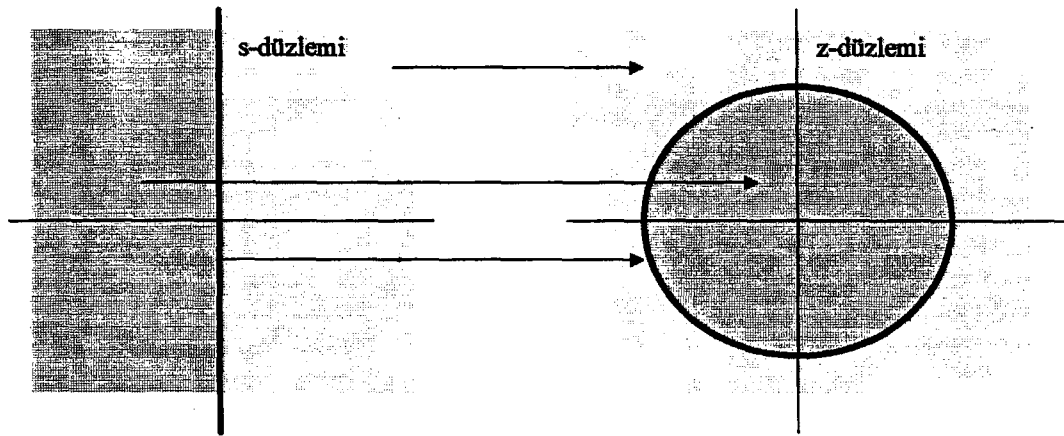
$$s = C. \frac{1 - z^{-1}}{1 + z^{-1}} \quad (2.30)$$

yazılmasıyla elde edilebilir. Bu transformun nasıl işlediği ve s -düzleminin z -düzlemine ne şekilde aktarıldığı Şekil 2.6' da iyice görülebilir.

Şekilden görülebileceği gibi s düzlemindeki $j\omega$ eksenini z -düzleminde birim çembere dönüştürmektedir. Bu özellik (2.29) denkleminin z için çözülüp;

$$z = \frac{(2/T) + s}{(2/T) - s} \quad (2.31)$$

s yerine $j\omega$ konulmasıyla bulunabilir.



Şekil 2.6 Bilineer z -dönüşümde s -düzleminin z -düzlemine transferi

$$z = \frac{(2/T) + j\omega}{(2/T) - j\omega} \quad (2.32)$$

$\omega=0$ iken $z=1$, $\omega=\infty$ iken $z=-1$ ve ω bu aralarda değişiyorken z , açısı 0 ile π arasında değişen değerler alır. s -düzlemindeki $j\omega$ ekseninin z 'te birim çembere transfer edildiği şartlar altında, tepkinin değişmediği dönüşüme ait (impulse invariant transform) tüm örnekleme hataları (aliasing errors) giderilir.

Analog frekans Ω ile dijital frekans ω arasında nonlineer bağıntı vardır. Bu nonlineerlik (2.30) denkleminin $s=j\Omega$ ve $z=e^{j\omega T}$ ifadeleriyle birlikte incelenmesiyle görülebilir.

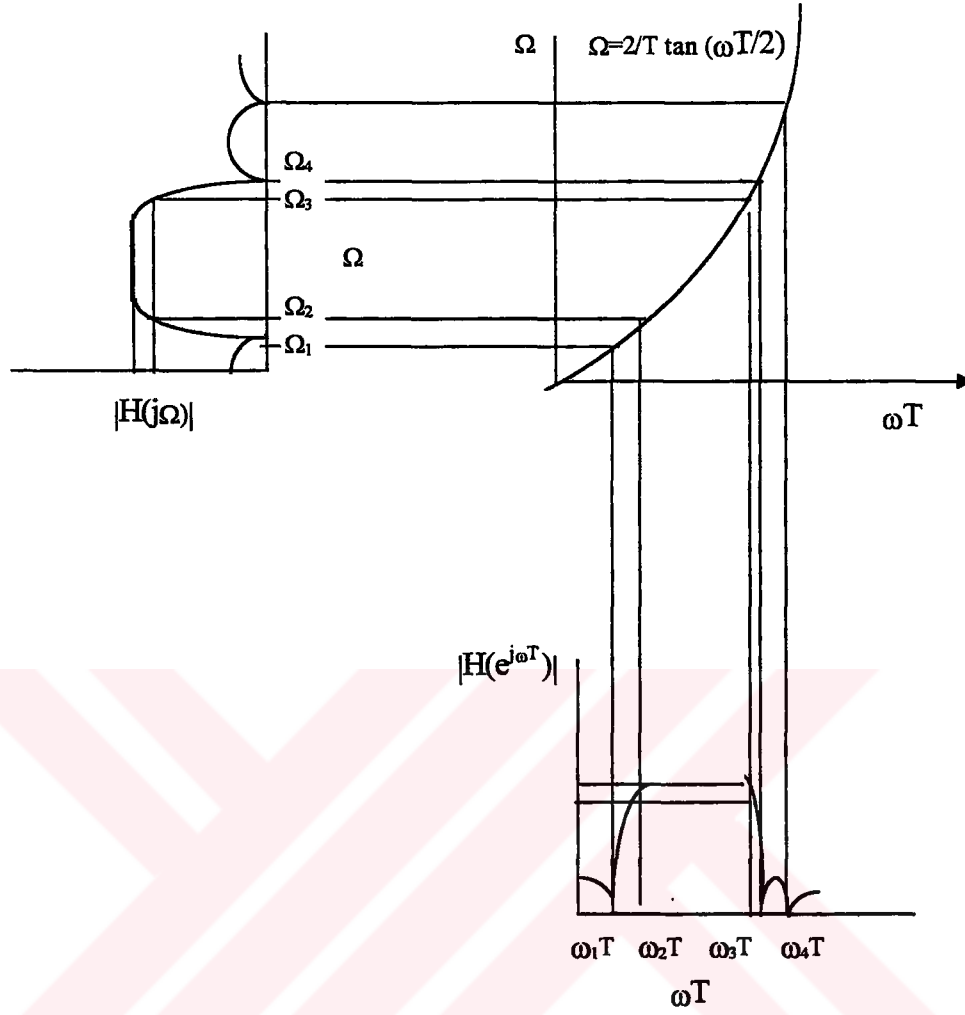
$$j\Omega = \frac{2}{T} \frac{(1 - e^{-j\omega T})}{(1 + e^{-j\omega T})} \quad (2.33)$$

Buradan Ω ;

$$\Omega = \frac{2}{T} \tan\left(\frac{\omega T}{2}\right) \quad (2.34)$$

elde edilir.

Küçük ω değerleri için transfer lineerdir. Fakat çoğu frekans ölçeklemesi için transfer iyice nonlineerdir. Bilineer dönüşüm kullanıldığında bu etki kötü sınırlamalar getirir. Bu demektir ki analog filtrenin genlik cevabı sabit parçalara bölünmüş olmalıdır. Eğer böyle yapılmazsa dijital frekans cevabı çarpıtılır ve bozulur. Bu frekans bozulmasını gideren kompanzasyon filtreleri vardır. Kompanzasyon tekniğini gösteren yapı Şekil 2.7' de gösterilmiştir.



Şekil 2. 7 Nonlinear Frekans Bozulmalarının Kompanze Edilmesi

Bu teknikte, dijital filtrenin istenen kesim frekansları Şekil 2.7' de sağ alt kısımda belirlenir. Bu örnekte olduğu gibi dört frekansımız (w_1, w_2, w_3, w_4) olsun. Dijital ve analog frekans ölçeklemelerindeki frekans bozulması ile ilgili (2.34) bağıntısından, filtrenin kesim frekansları yeni analog kesim frekanslarına ($\Omega_1, \Omega_2, \Omega_3, \Omega_4$) çevrilir. Sonuçta Şekil 2.7' de sol üstte görülen, kesim frekansları uygun şekilde biçimlendirilmiş analog filtre elde edilir. Bu analog filtreye bilinear dönüşüm uygulanarak istenen kesim frekanslarına sahip dijital filtre elde edilir.

Özet olarak bilinear-z dönüşümü, sürekli ve dijital filtre fonksiyonları arasında s-düzlemin sol yarı bölgesini z-düzleminde birim çember içerisine transfer eden basit bir dönüşüm olarak niteliyebiliriz. Gerçeklenebilir kararlı sürekli yapılar gerçekleştirilebilir kararlı dijital sistemlere dönüştürülür. Band genişliği keskin olan filtre yine band genişliği keskin diğer yapıya frekans tepkisinde herhangi bir bozulma olmadan transfer edilir. Tek kötü yanı, dijital ve analog frekanslar arasında nonlineer bağıntıya sahip olmasıdır. Bu yüzden sürekli sistemin frekans tepkisi, kompanze edilecek şekilde küçük sabit yapıya bölünmelidir. Ayrıca analog filtrenin faz tepkisi ve impulse tepkisi dijital filtreye bozuk şekilde aktarılmaktadır.

3. DİJİTAL FİLTRENİN GERÇEKLENMESİ

Birinci bölümden takip edilirse, üç aşamada filtrenin oluşturulacağı görülebilir.

Bunlar;

- 1) Analog sinyalin örneklenmesi ve sayısal forma çevrilmesi;
- 2) Sayısal formdaki bilginin filtre yapısına göre çeşitli algoritmalara (burada 80C32'de filtre programlarına) tabi tutulması;
- 3) Filtrelenen değerın tekrar analog sinyale dönüştürülmesidir.

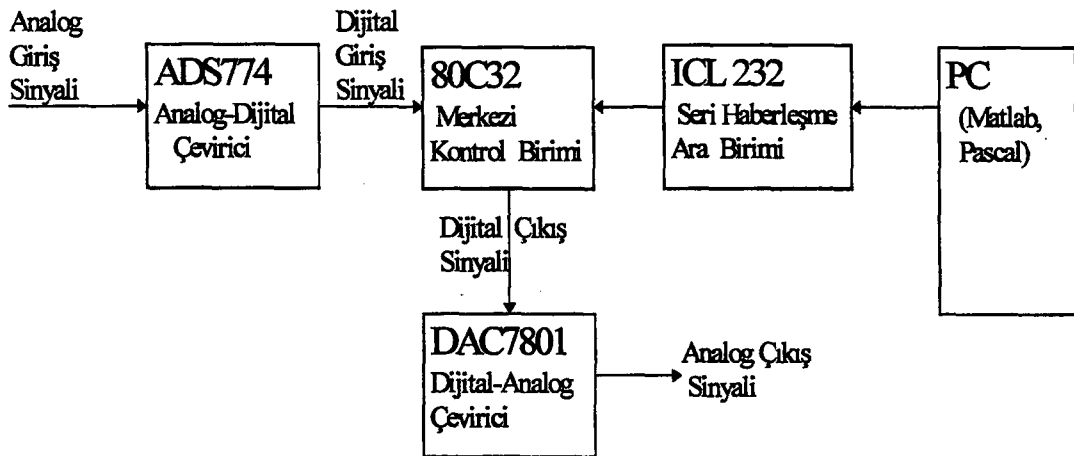
Gerçekleştirdiğim bitirmede, analog sinyali örnekleyip sayısal bilgiye çeviren yapı, Analog Dijital Çevirici (ADC); belirlenen filtre karakteristiklerini sağlayacak şekilde oluşturulan alt programları içeren ve bu algoritmalara göre örneklenmiş sinyali işleyen mikrokontrolör; ve sonuç değeri (filtre edilmiş dijital bilgiyi) analog sinyale çeviren yapı, Dijital Analog Çevirici (DAC); tek bir kart üzerinde gerçekleştirilmiştir.

Analog sinyalin örneklenmesinde 12 bitlik ADS774 A/D çevirici kullanılmıştır. Mikroişlemci uyumlu olan bu entegre devre (IC), Bipolar çıkış verecek şekilde (1 bit işaret biti) donanımda yerleştirilmiştir. Böylece $\pm 10V$ arasındaki analog sinyaller

örneklenebilmektedir. $8.5\mu\text{sn}$ 'lik çevrim zamanına sahiptir. Dijital bilgiyi analog sinyale çevirmede yine 12 bitlik DAC7801 D/A çevirici entegresi kullanılmıştır. A/D çevirici gibi bu entegre bipolar çıkış verecek şekilde ($\pm 10\text{V}$) donanımsal olarak ayarlanmıştır. Her iki entegreye mikroişlemci uyumlu olmalarından dolayı yazılımla rahatça erişilip gerekli düzenlemelerde bulunulabilmektedir.

İşlemci olarak 80C32 mikrokontrolör entegre devresi kullanılmıştır. 8 bitlik veri ve 16 bitlik adres yoluna sahiptir. 3 tane 16 bitlik sayıcı/zamanlayıcısı, 32 bit i/o ucu, full duplex seri portu ve entegre içinde saat devresi vardır. Ayrıca 256 byte'lık dahili RAM'İ vardır. Filtreleme işlemini gerçekleştiren program ve alt programlar 16k'lık EPROM'a yazılmıştır. Bu program, dışarıdan filtre karakteristiklerine müdahale edebilecek şekilde esnek bir yapıda oluşturuldu. Programda kullanılan filtre katsayılarının bulunması PC'de Matlab ile gerçekleştirilmektedir. Matlab'teki program ile istenen karakteristiğe sahip filtre katsayıları hesaplanmaktadır. Katsayıların, filtreleme işlemlerinin gerçek zamanda gerçekleştirildiği karta aktarılması ise Pascal programı ile 80C32'nin seri haberleşme özelliğinden yararlanılarak seri yapıda aktarılmaktadır. ICL232 entegresi bu amaçla kullanılmıştır.

Bu anlatılan sistemin genel yapısı aşağıdaki şekilde gösterilmiştir.



Şekil 3.1 Dijital Filtrenin Blok Diagramı

3.1 : DİJİTAL FİLTRE DİZAYNI

Bölüm 2 ile elde edilen dijital filtre fonksiyonu genel hali ile

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{i=0}^n a_i \cdot z^{-i}}{\sum_{i=0}^n b_i \cdot z^{-i}} = K \frac{\prod_{i=0}^n (z - z_i)}{\prod_{i=0}^n (z - p_i)} \quad (3.1)$$

şeklinde yazılır. Transfer fonksiyonunun pay ve payda fonksiyonu dereceleri gerçekleştirilebilirlik şartı gözönünde bulundurulularak dereceleri eşit alınmıştır. Formülden elde edilecek fark denklemleri, dijital filtrenin bilgisayar mimarisinde oluşturulacak programı için verinin işleme şeklini, dolayısıyla filtreleme programını vermektedir. (3.1)' in gerçekleşmesi için temelde 4 tane yapı vardır. (1D, 2D, 3D, ve 4D). Birde katsayı hassasiyetini gidermek için bu 4 yapı ayrıca ardışıl, paralel ve merdiven formda da gerçekleştirilebilir.

3.2 DOĞRUDAN GERÇEKLEMELER

3.2.1: 1D YAPISI (BİRİNCİ DOĞRUDAN GERÇEKLEME)

(3.1) denklemini aşağıdaki gibi düzenlenirse;

$$H(z) = \frac{Y(z)}{M(z)} \frac{M(z)}{X(z)} = \frac{\sum_{i=0}^n a_i \cdot z^{-i}}{\sum_{i=0}^n b_i \cdot z^{-i}} \quad (3.2)$$

Y(z) ve X(z) ayrı ayrı fonksiyonlar olarak denklemden çekilirse;

$$\frac{Y(z)}{M(z)} = \sum_{i=0}^n a_i \cdot z^{-i} \quad (3.3)$$

$$\frac{X(z)}{M(z)} = \sum_{i=0}^n b_i \cdot z^{-i} \quad (3.4)$$

eşitliklerini yazabiliriz. Ardından çıkış için ara geçiş değerlerini veren M(z) elde edilir.

$$X(z) = \sum_{i=0}^n b_i \cdot z^{-i} \cdot M(z) \quad (3.5)$$

$$M(z) = X(z) - \sum_{i=1}^n b_i \cdot z^{-i} \cdot M(z) \quad (3.6)$$

Benzer şekilde çıkış fonksiyonu Y(z);

$$Y(z) = \sum_{i=0}^n a_i \cdot z^{-i} \cdot M(z) \quad (3.7)$$

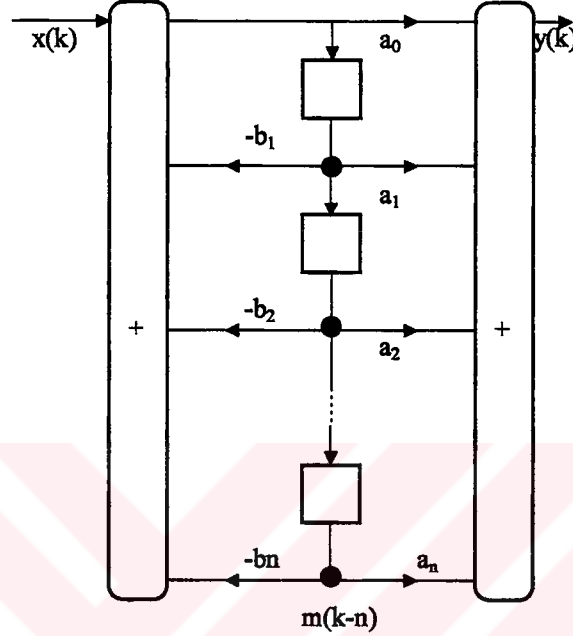
bulunur. Buradan kesikli-zamana (discrete-time) geçilirse;

$$m(k) = x(k) - \sum_{i=1}^n b_i \cdot m(k-i) \quad (3.8)$$

ve

$$y(k) = \sum_{i=0}^n a_i \cdot m(k-i) \quad (3.9)$$

elde edilir. (3.8) ve (3.9) denklemleri aşağıda şekli görülen 1D yapısını tanımlar.



Şekil 3.2 1D Dijital Filtre Gerçeklemesi

Şekil 3.2' de gecikme elemanları (z^{-1}) dikdörtgen kutularla, çarpma işlemleri ok ve üzerindeki değerlerle, toplama işlemleri + içeren uzun dikdörtgenlerle ve sinyal dağıtım noktaları siyah büyük noktalarla gösterilmiştir. 1D yapısı n.ci dereceden bir filtre için sadece n tane gecikme elemanı içerdiği için kanonik yapı olarak isimlendirilir.

3.2.2: 2D YAPISI (İKİNCİ DOĞRUDAN GERÇEKLEME)

Bu yapı 1D gerçeklemesinin transpozunun alınmasıyla elde edilir. Bu işlemde sinyal akış yönü değiştirilir. Bu durumda toplama noktaları sinyal dağıtım noktalarına, giriş noktaları çıkışa dönüşmektedir. Transpozu alınmış bu devrenin transfer fonksiyonu orjinal devreyle aynıdır.

Bu şekilde yapılan deęişlikle elde edilen 2D yapısı Şekil 3.3 ile gösterilmiştir.

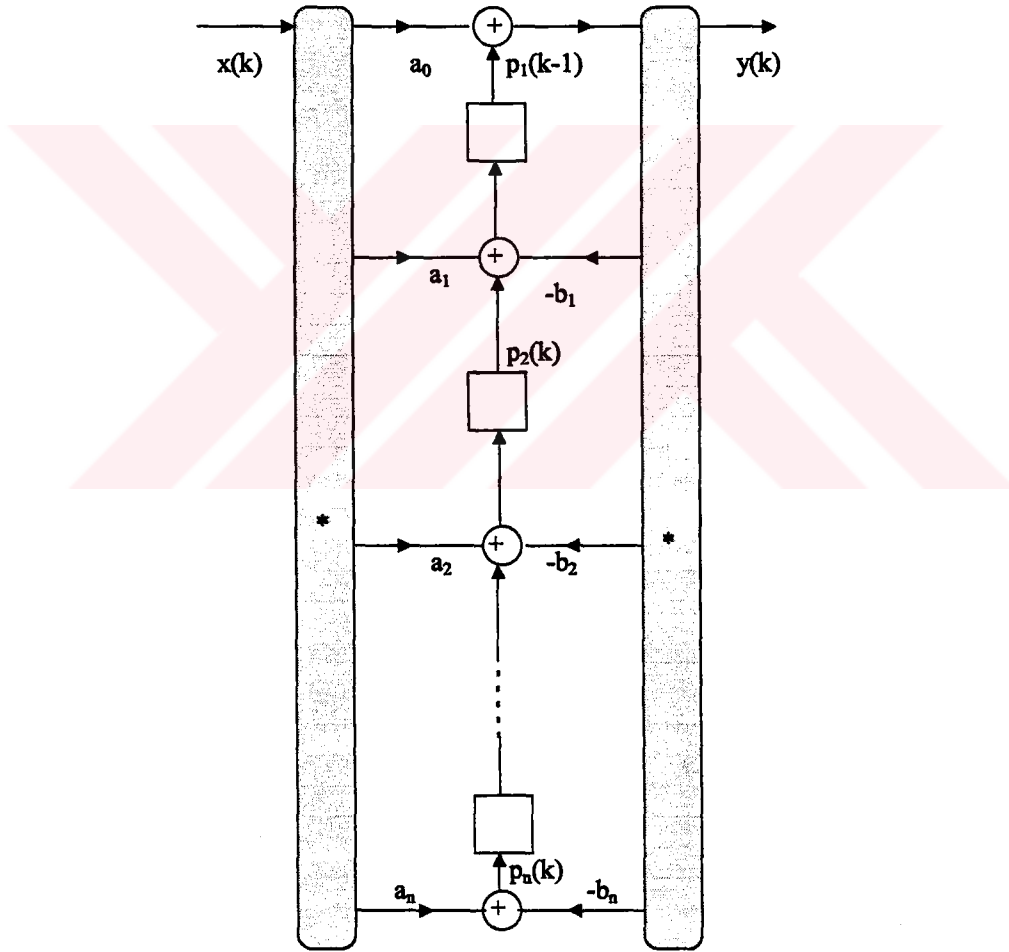
2D fark denklemleri ise ;

$$p_i(k-1) = p_{i+1}(k-1) + a_i \cdot x(k) - b_i \cdot y(k-1) \quad i=1, n-1 \quad (3.10)$$

$$p_n(k) = a_n \cdot x(k) - b_n \cdot y(k) \quad (3.11)$$

$$y(k) = a_0 \cdot x(k) + p_1(k-1) \quad (3.12)$$

şeklinde olan bu yapıda kanoniktir.



Şekil 3.3 2D Dijital Filtre Gerçeklemesi

3.2.3: 3D YAPISI (ÜÇÜNCÜ DOĞRUDAN GERÇEKLEME)

(3.1) denklemini aşağıdaki gibi düzenlersek;

$$Y(z) \cdot \sum_{i=0}^n b_i \cdot z^{-i} = X(z) \cdot \sum_{i=0}^n a_i \cdot z^{-i} \quad (3.13)$$

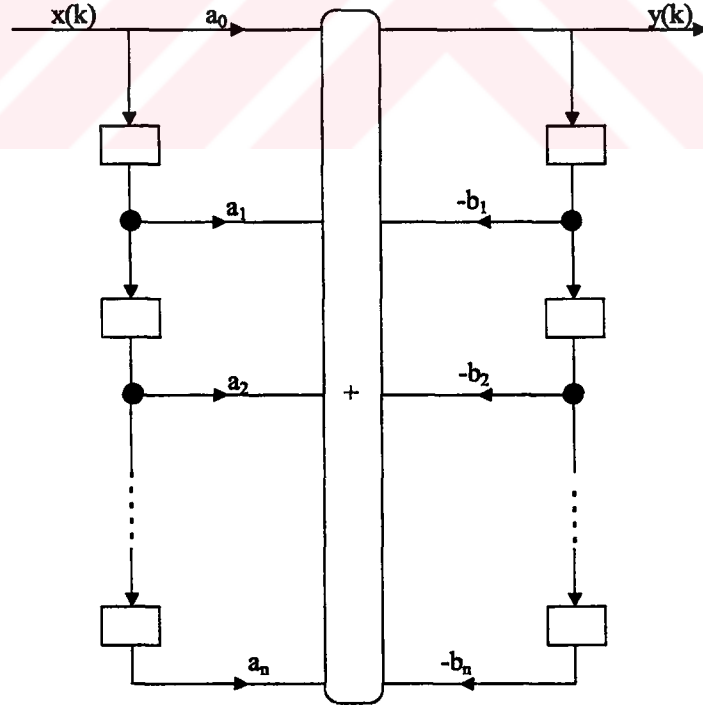
ve $Y(z)$ buradan çekilirse;

$$Y(z) = \sum_{i=0}^n a_i \cdot z^{-i} \cdot X(z) - \sum_{i=1}^n b_i \cdot z^{-i} \cdot Y(z) \quad (3.14)$$

olarak elde edilir. Kesikli-zamana (Discrete-time) geçilirse sonuç fark denklemi;

$$y(k) = \sum_{i=0}^n a_i \cdot x(k-i) - \sum_{i=1}^n b_i \cdot y(k-i) \quad (3.15)$$

elde edilir. Bu yapı Şekil 3.4' te gösterilmiştir.

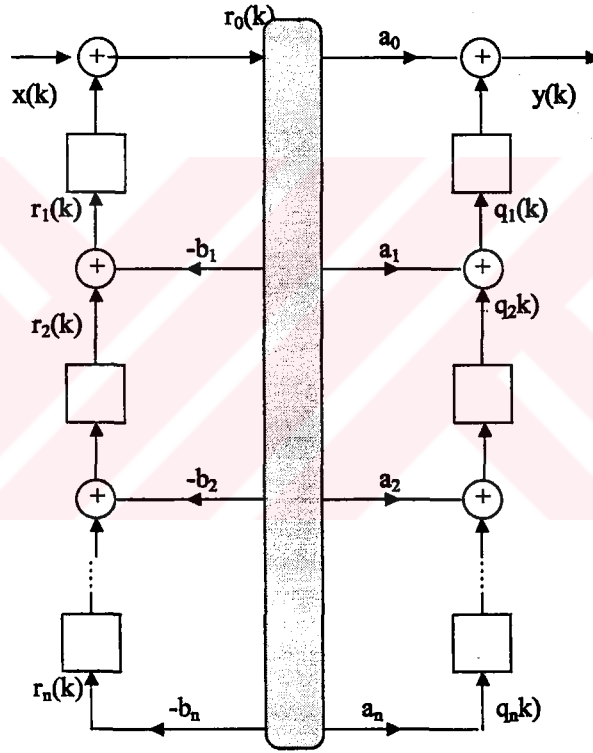


Şekil 3.4 3D Dijital Filtre Gerçeklemesi

Bu yapı sadece bir toplama noktası içerir. Fakat $2n$ gecikme elemanına sahiptir.

3.2.4: 4D YAPISI (DÖRDÜNCÜ DOĞRUDAN GERÇEKLEME)

4D yapısı 3D yapısının transpozudur. Bu yapının blok diagramını Şekil 3.5' te gösterilmiştir.



Şekil 3.5 4D Dijital Filtre Gerçeklemesi

Bu yapı sadece bir sinyal dağıtım noktasına sahiptir. Fakat $2n$ tane fark denklemine sahiptir.

$$r_0(k) = x(k) + r_1(k-1) \quad (3.16)$$

$$q_n(k) = a_n \cdot r_0(k) \quad (3.17)$$

$$r_n(k) = -b_n \cdot r_0(k) \quad (3.18)$$

$$q_i(k) = a_i \cdot r_0(k) + q_{i+1}(k-1) \quad i=1, n-1 \quad (3.19)$$

$$r_i(k) = -b_i \cdot r_0(k) + r_{i+1}(k-1) \quad (3.20)$$

$$y(k) = a_0 \cdot r_0(k) + q_1(k-1) \quad (3.21)$$

Bu dört yapının sahip olduğu özellikler Tablo 3.1'de gösterilmiştir.

	Model			
	1D	2D	3D	4D
Zaman Gecikme Elemanları	n	n	2n	2n
Çarpma Elemanları	2n+1	2n+1	2n+1	2n+1
Toplama Noktaları	2	n+1	1	2n
Sinyal Dağıtım Noktaları	n+1	2	2n	1

Tablo 3.1 Doğrudan Gerçeklemelerin Özellikleri

3.3 ARDIŞIL (CASCADE) GERÇEKLEMELER

Katsayı hassaslığının getirdiği problemlerden kaçınmak için (3.1)'deki $H(z)$ filtre transfer fonksiyonu ikinci dereceden modüllerle ve bu modüllerin ardışıl bağlanmalarıyla gerçekleştirilir.

$$H(z) = \frac{\prod_{i=1}^m (\alpha_{i0} + \alpha_{i1} \cdot z^{-1} + \alpha_{i2} \cdot z^{-2})}{\prod_{i=1}^m (1 + \alpha_{i3} \cdot z^{-1} + \alpha_{i4} \cdot z^{-2})} \quad (3.22)$$

Bu denklemde $m, n/2'$ den küçük en büyük veya eşit tam sayı olarak seçilir. Eğer pay ve payda çarpanları eşleştirilmiş ve modüller sıralı dizilmiş ise , $H(z)$;

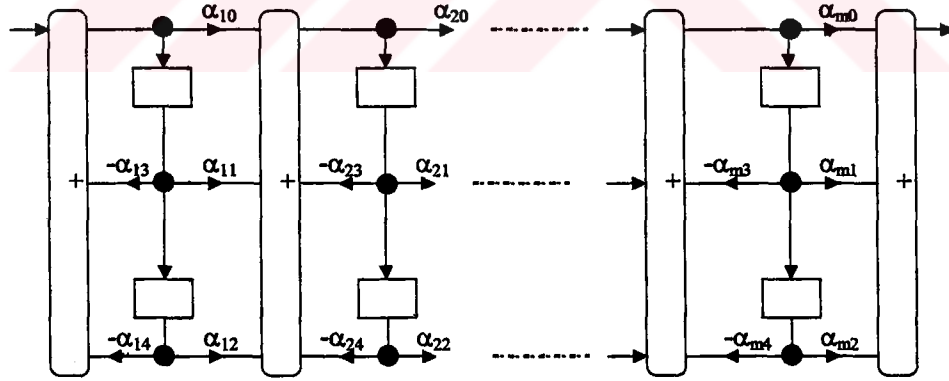
$$H(z) = \prod_{i=1}^m A_i(z) \quad (3.23)$$

şeklinde ve A_i' ler;

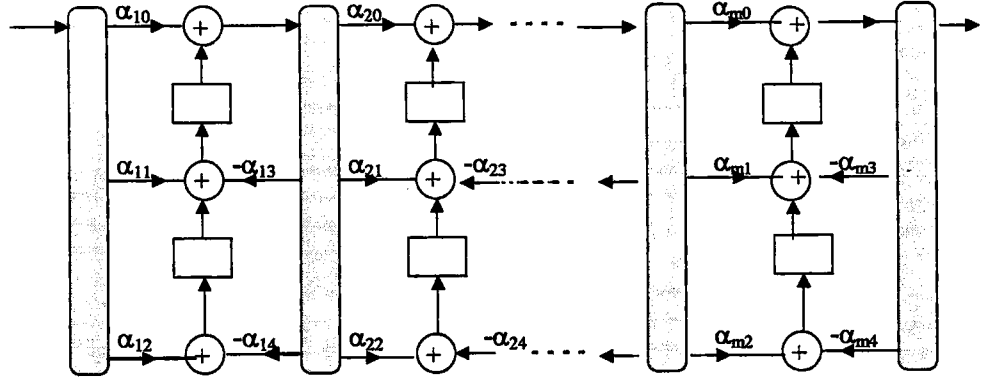
$$A_i(z) = \frac{(\alpha_{i0} + \alpha_{i1} \cdot z^{-1} + \alpha_{i2} \cdot z^{-2})}{(1 + \alpha_{i3} \cdot z^{-1} + \alpha_{i4} \cdot z^{-2})} \quad (3.24)$$

olarak ifade edilmişlerdir. Bu ikinci dereceden modüller arka arkaya bağlanarak (Şekil 3.6' daki gibi) filtre transfer fonksiyonu katsayı hassaslığı giderilmiş olarak gerçekleştirilir.

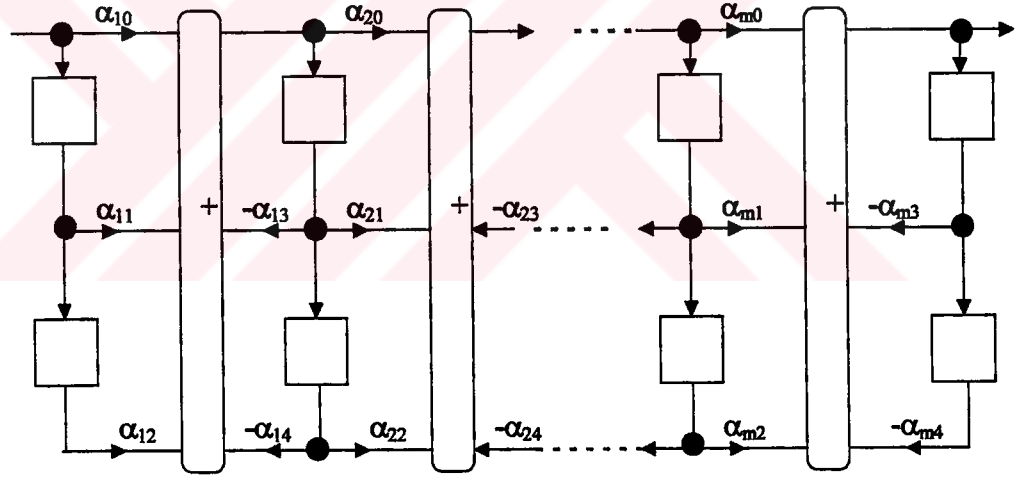
Bu ikinci dereceden modüller doğrudan gerçekleştirme yapılarından birisiyle gerçekleştirilir. Aşağıdaki şekillerde 1D, 2D, 3D ve 4D dijital filtre doğrudan gerçekleştirme yapılarıyla gerçekleştirilmiş ardışıl (cascade model) dijital filtre yapıları sıralanmıştır.



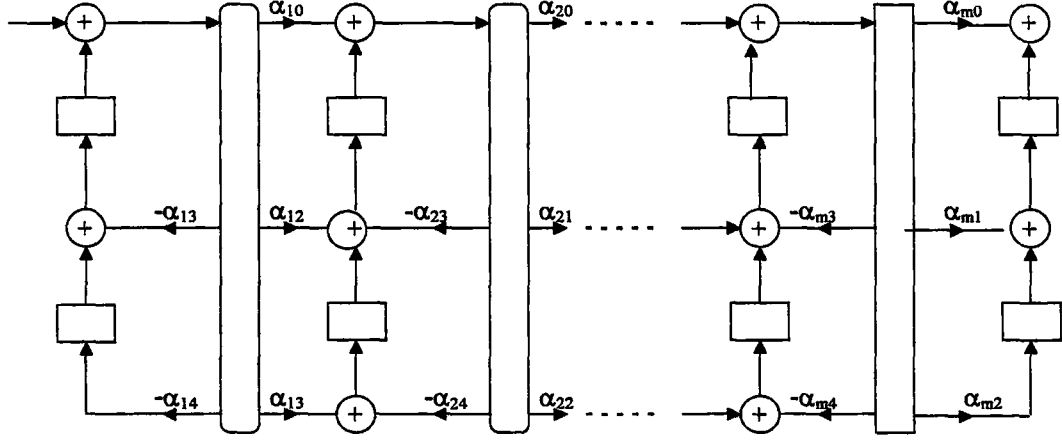
Şekil 3.6 1D Yapısıyla Gerçekleştirilmiş Ardışıl Dijital Filtre Yapısı



Şekil 3.7 2D Yapısıyla Gerçekleştirilmiş Ardışıl Dijital Filtre Yapısı



Şekil 3.8 3D Yapısıyla Gerçekleştirilmiş Ardışıl Dijital Filtre Yapısı



Şekil 3.9 4D Yapısıyla Gerçekleştirilmiş Ardışıl Dijital Filtre Yapısı

Bu ardışıl yapılar Tablo 3.2' de karşılaştırılmıştır. Tablo 3.1 ve 3.2 karşılaştırıldığında, ardışıl 3D ve 4D yapıları her iki durumda da $n-2$ gecikme elemanına sahip olmaktadır. Ardışıl yapı her durum için extra çarpma elemanları getirmektedir. Ardışıl hale getirilmiş doğrudan yapılar, normal doğrudan yapılara göre $m-1$ tane extra toplama noktası ve sinyal dağıtım noktası yapıya katmaktadır.

	Model			
	1D	2D	3D	4D
Gecikme	$2m$	$2m$	$2m+2$	$2m+2$
Elemanı	(n)	(n)	$(2n-(n-2))$	$(2n-(n-2))$
Çarpma	$5m$	$5m$	$5m$	$5m$
Elemanı	$(n+n+m)$	$(2n+m)$	$(2n+m)$	$(2n+m)$
Toplama	$m+1$	$3m$	m	$3m+1$
Noktaları	$(2+m-1)$	$(n+m)$		$(2n-(m-1))$
Sinyal Dağıtım	$3m$	$m+1$	$3m+1$	m
Noktaları	$(n+m)$	$(2+m-1)$	$(2n-(m-1))$	

Tablo 3.2 Ardışıl Yapıda Eleman Sayıları

- $m, n/2$ 'den küçük en büyük veya ona eşit tam sayıdır. Parantez içindeki sayılar Tablo 3.1 ile olan karşılaştırmayı verir.

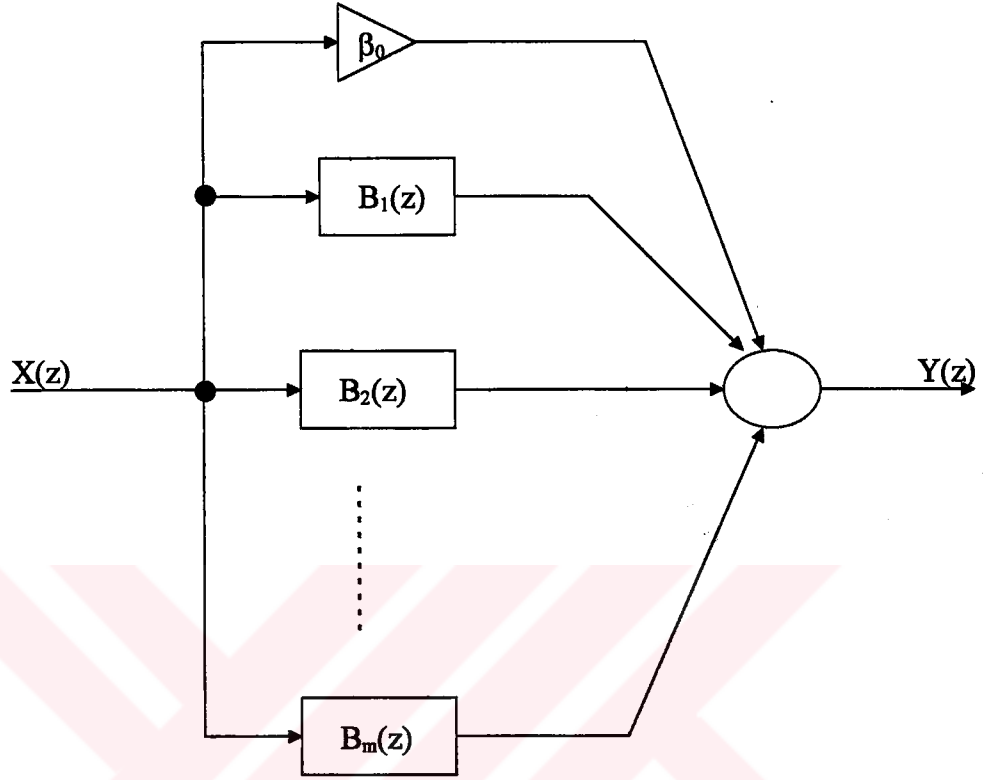
3.4 PARALEL GERÇEKLEME

Katsayı hassasiyetinin getirdiği problemlerden kaçınmanın diğer bir yolu, $H(z)$ transfer fonksiyonunun paydasının çarpanlara ayrılması ve kısmi-çarpanlara ayırma metodu ile (partial-fraction expansion) filtre fonsiyonun gerçekleştirilmesidir.

$$H(z) = \beta_0 + \sum_{i=1}^m B_i(z) \quad (3.25)$$

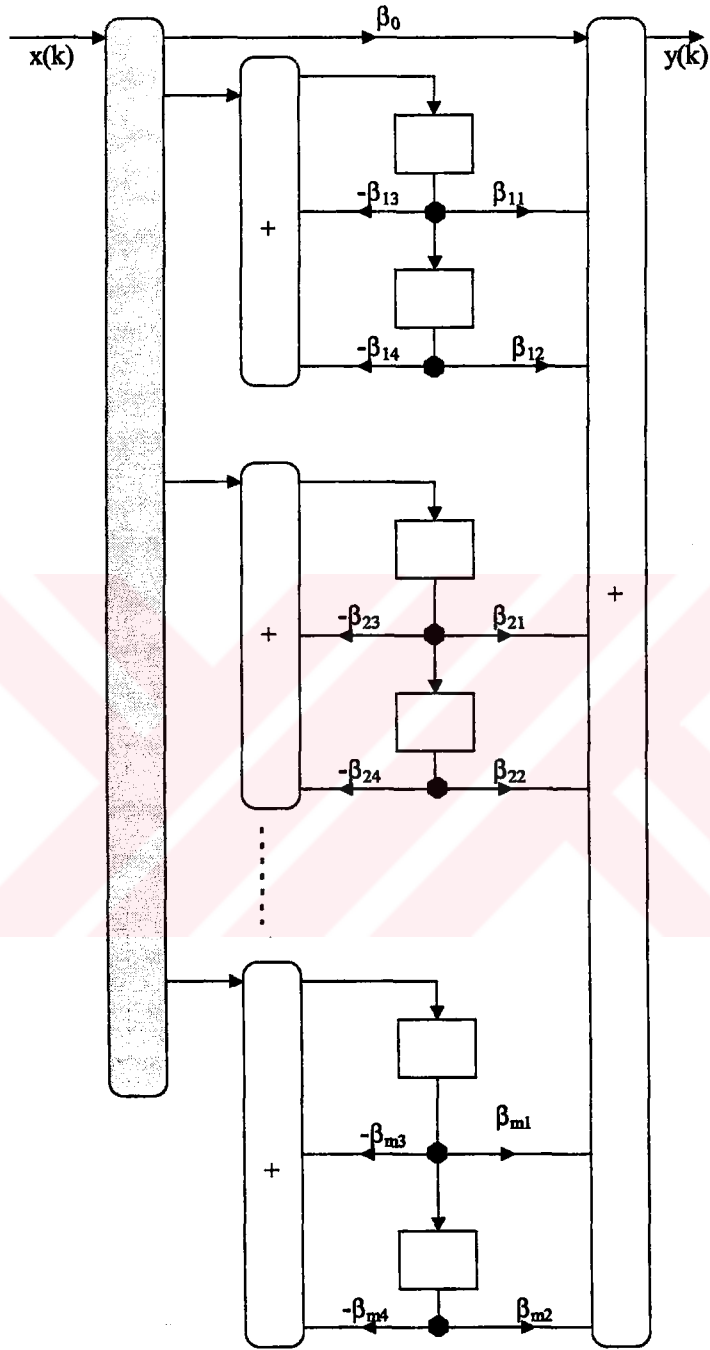
$$B_i(z) = \frac{\beta_{i1} \cdot z^{-1} + \beta_{i2} \cdot z^{-2}}{1 + \beta_{i3} \cdot z^{-1} + \beta_{i4} \cdot z^{-2}} \quad (3.26)$$

(3.26)'daki fonsiyonların toplamı haline getirilen $H(z)$, Şekil 3.10'da gösterildiği gibi paralel yapıda gerçekleştirilebilir.

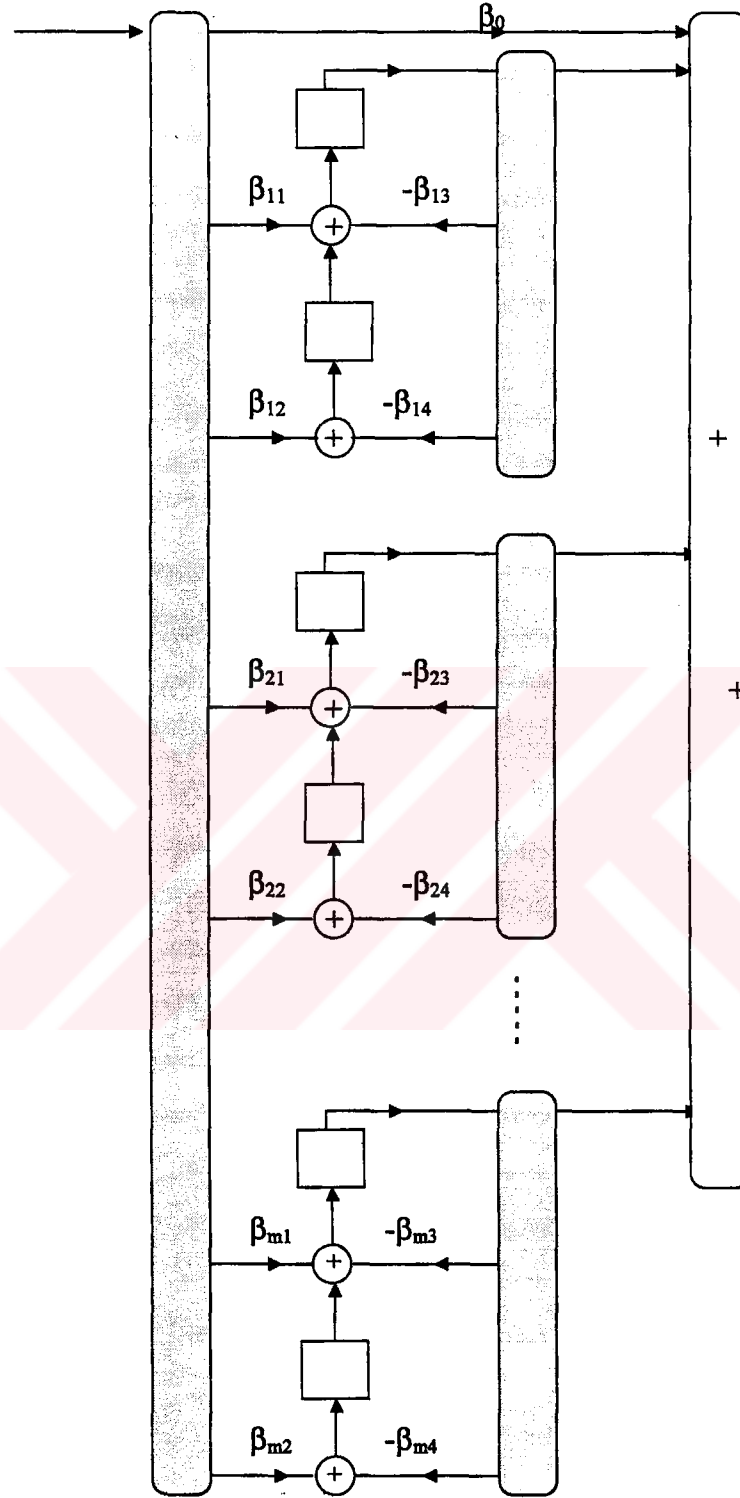


Şekil 3.10 Dijital Filtrenin Paralel Gerçeklenmesi

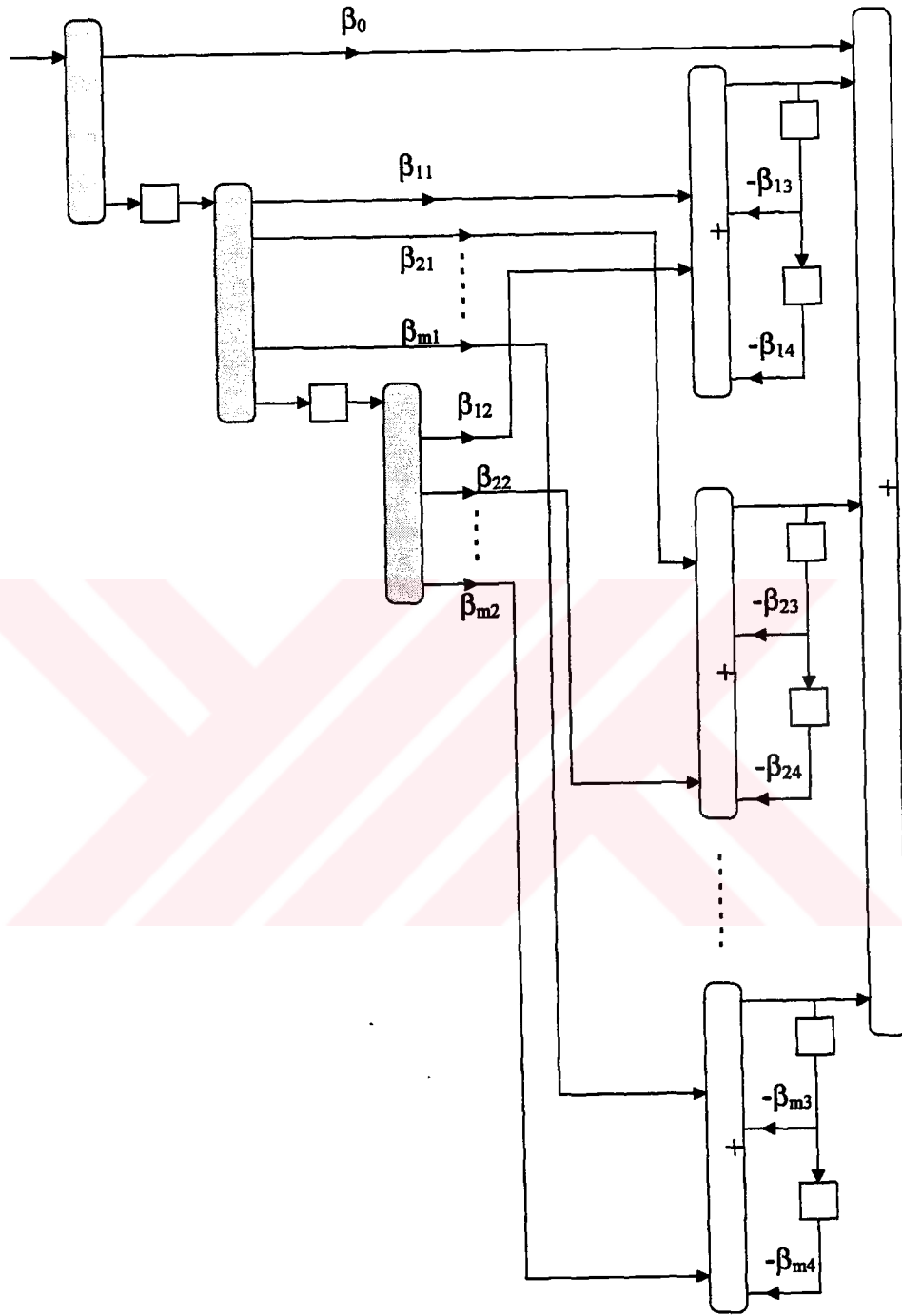
Bölüm 3.2' de anlatılan doğrudan (Direct Realization) gerçekleştirme yöntemlerinden birisiyle (3.26)'daki fonksiyonları içeren paralel yapılar gerçekleştirilir. Bu gerçeklemeler aşağıdaki şekillerde sıralanmıştır.



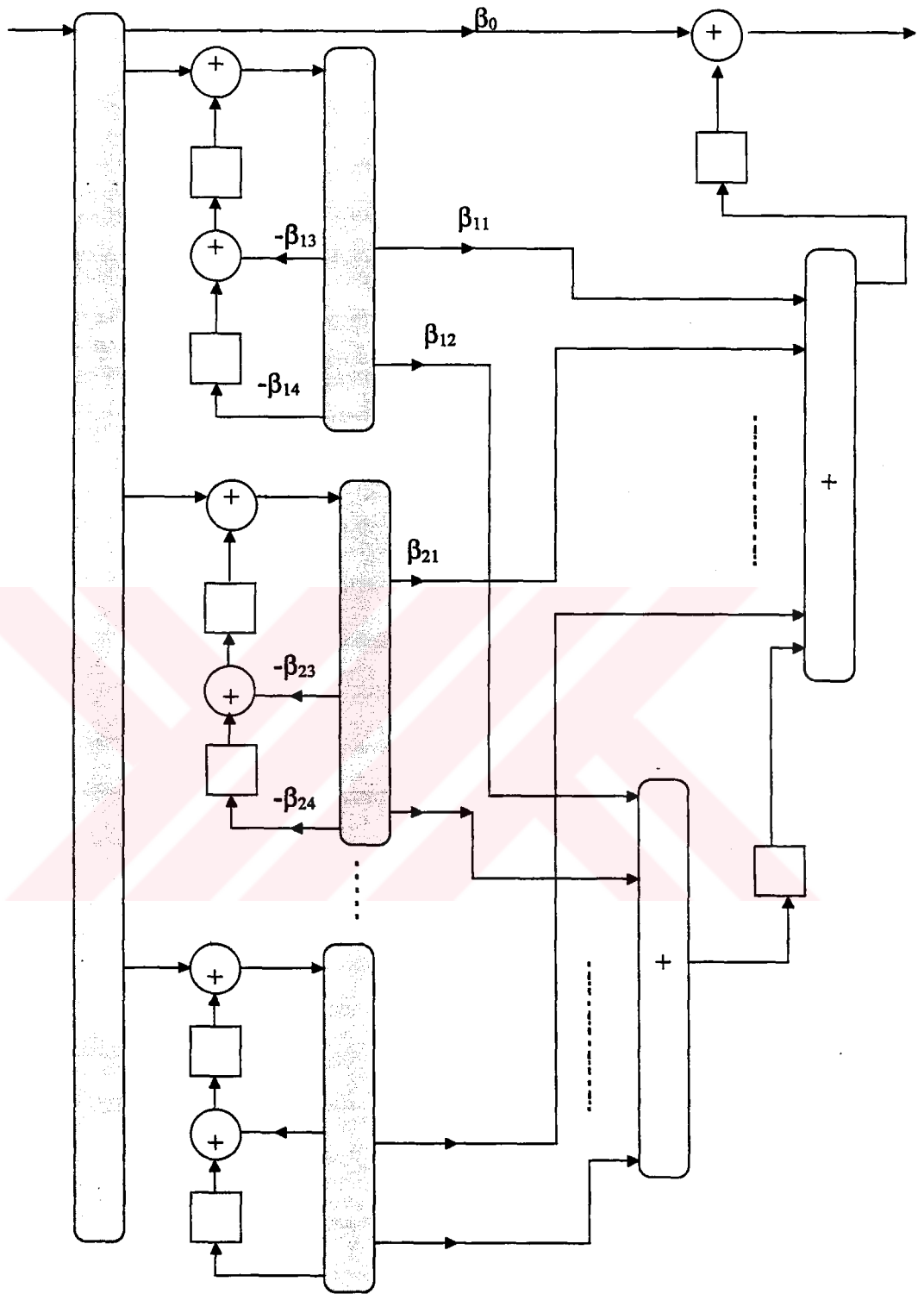
Şekil 3.11 1D İle Gerçekleştirilen İkinci Dereceden Paralel Yapılar



Şekil 3.12 2D İle Gerçekleştirilen İkinci Dereceden Paralel Yapılar



Şekil 3.13 3D İle Gerçekleştirilen İkinci Dereceden Paralel Yapılar



Şekil 3.14 4D İle Gerçekleştirilen İkinci Dereceden Paralel Yapılar

Tablo 3.3 doğrudan yapılarla gerçekleştirilen paralel dijital filtre karakteristiklerinin karşılaştırmasını yapmaktadır.

	Model			
	1D	2D	3D	4D
Gecikme	2m	2m	2m+2	2m+2
Elemanı	(n)	(n)	(2n-(n-2))	(2n-(n-2))
Çarpma	4m+1	4m+1	4m+1	4m+1
Elemanı	(2n+1)	(2n+1)	(2n+1)	(2n+1)
Toplamı	m+1	2m+1	m+1	2m+3
Noktaları	(2+m-1)	(n+1)	(m+1)	(2n-(n-3))
Sinyal Dağıtım	2m+1	m+1	2m+3	m+1
Noktaları	(n+1)	(2+m-1)	(2n-(n-3))	(m+1)

Tablo 3.3 Paralel Gerçekleme Eleman Sayıları

* m, n/2'den küçük en büyük veya ona eşit tam sayıdır. Parantez içindeki sayılar Tablo 3.1 ile olan karşılaştırmayı verir.

Tablo 3.3 incelendiğinde 2D ve 4D yapılarının doğrudan gerçeklemeye göre n-2 gecikme elemanını koruduğu görülür. Çarpma elemanlarının sayısının Tablo 3.1 ile aynıdır. 1D paralel yapısı (m-1) ek toplama noktası içerirken, 2D yapısında fazladan m-1 sinyal dağıtım noktası gerektirmektedir. 3D yapısı (m) adet ek toplama noktası gerektirirken n-3 tane daha az sinyal dağıtım noktası içerir.

4. MİKROKONTROLÖR İLE (80C32) DİJİTAL FİLTRENİN GERÇEKLENMESİ

Önceki bölümlerde anlatılan tekniklerin uygulanmasıyla elde edilen sayısal filtre fonksiyonun 80C32 mikrokontrolörüyle gerçekleştirilmesinde 1D (first direct structure) doğrudan sayısal filtre gerçekleştirilmesi metodu kullanıldı. Tablo 3.1 incelenirse 1D yapısının daha az sayıda toplama noktası ve zaman gecikme elemanı içerdiği görülür. Zaman-gecikme elemanlarının fazla olması mikrokontrolörde yazılan makina dili programının daha fazla döngü içermesine neden olacaktır. Buda daha fazla işlem ve örnekleme hızının düşmesi anlamına gelecektir. Aynı şekilde toplama noktalarının fazlalığında makina dili programının uzamasına ve dolayısıyla dijital filtrenin örnekleme hızının ve aralığının düşmesine neden olmaktadır.

4.1 SAYISAL FİLTRENİN DONANIM ÖZELLİKLERİ

80C32, 8 bitlik bir mikrokontrolördür. Veri yolu ile adres hatlarının düşük 8 biti aynıdır (multiplexed bus). Bu hatlardaki bilgi, kontrolör üzerindeki ALE (address latch enable) ucuyla (ALE=1 → adres, ALE= → veri) ayrılmaktadır. 256 byte'lık dahili RAM'i vardır. Bu RAM'in yüksek 256 byte'ı özel fonksiyon yazmaçlarını (portları, interrupt ve zamanlayıcı/sayıcı yazmaçlarını v.b.) içerir. 64K'lık harici program ve veri bellek alanına sahiptir. 32 tane giriş çıkış hattı vardır. Bu hatlar aynı zamanda adres, veri ve kontrol sinyallerinin üretildiği uçlardır. Full-duplex seri haberleşme özelliği vardır. Dahili osilatör ve saat devresi vardır. Dışarıdan sadece kristal bağlanılarak entegrenin saat sorunu çözülür. Aritmetik işlemlerini işaretsiz olarak gerçekleştirir.

İşaretlerin işlenmesinde 12 bitlik ADC774 ve DAC7801 kullanılmıştır. Unipolar ve Bipolar sinyal işleme (negatif genliğe sahip sinyalleri örnekleme veya negatif genlikli

sinyal üretebilme) özelliklerine sahiptirler. Gerçekleştirilen kartta her iki entegrede bipolar olarak ve $\pm 10V$ aralığında çalışacak şekilde ayarlanmıştır. ADC774 analog-dijital çevirici elemanı, 12 bitlik örnekleme için 7.5μ saniyelik çevirim süresine sahiptir. Yani yaklaşık 133 KHz'lik örnekleme frekansına sahiptir. Ancak 80C32 nin yavaş kalmasından dolayı (bir komut çevrim süresi $\approx 0.85\mu$ saniyedir.) filtre alt programının işleme süresi bu örnekleme hızını düşürmektedir (5 KHz). Bu yüzden örnek alınmasında interrupt alt programına gerek görülmemiştir. Dijital-analog çevirici olarak kullanılan DAC7801 400n saniye çıkış akımı sabitleme süresine sahiptir. DAC7801 iki adet dijital-analog çevirici içerir ve ikiside akım çıkışlı olma özelliğine sahiptir. Bu akım iki opamp ile yükseltilerek gerilime çevrilmiştir. Mikroişlemci uyumlu olan entegreler doğrudan mikrokontrolörün veri ve kontrol yollarına bağlanmışlardır. Böylece yazılımla denetlenip veri aktarım işlemleride kolayca gerçekleştirilmiştir.

80C32'in yetersizliğinden dolayı sayısal filtre katsayıları PC'de matlab ile hesaplanmış ve Pascal programı ile de 80C32'nin seri özelliği kullanılarak mikrokontrolöre aktarılmıştır. Katsayı aktarımı için ICL 232 entegresi kullanıldı. Çift yönlü RS-232 alıcı verici ara devresi olan entegre bilgilerin +10V ve -10V'luk sinyaller olarak iletilip alınmasını sağlar. Kontrolör ile 9600 baud rate'te alma işlemi yapılmış, bu oran için 80C32'nin timerları kullanılıp sayısal filtrenin pay ve payda polinomu katsayıları bilgisayardan alınmıştır.

4.2 SAYISAL FİLTRE YAZILIMININ AÇIKLANMASI

80C32 ile yazılan sayısal filtre programı başlıca 4 kısma ayrılabilir. Bunlar;

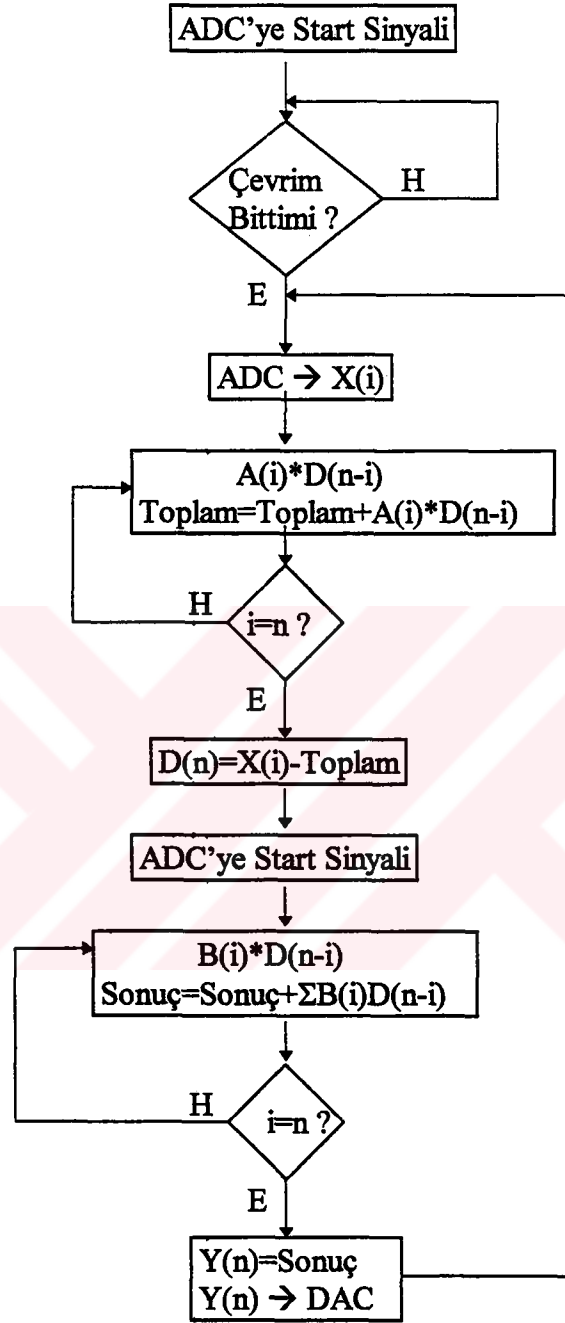
- Dijital filtre katsayılarının en başta seri haberleşme ile PC'den alınması ve bunların filtreleme programında kullanılan adreslere uygun formatta kaydedilmeleri.
- ADC774'ün kontrol edilmesi ve düzenli aralıklarla örneklenmiş sinyalin hafızaya yazılması.

- DAC7801'e her filtreleme algoritmasının bitiminden sonra filtrelenmiş verinin gönderilmesi ve analog çıkış sinyalinin elde edilmesi.
- 1D sayısal filtre algoritmasının gerçekleştirilmesi.

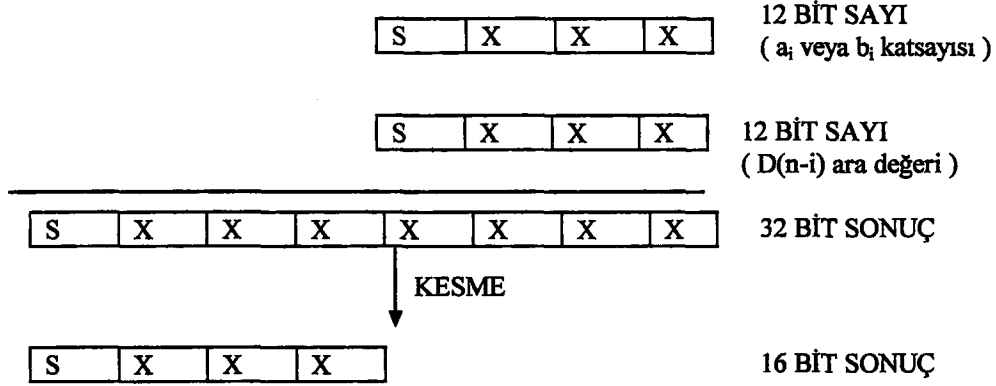
Denklem (3.8) ve (3.9) ile kesikli-zaman bağıntıları verilen ve Şekil 3.2 ile grafi gösterilen 1D yapısının akış diagramı Şekil 4.1'de verilmiştir. Şekilden görüldüğü gibi sayısal filtrenin derecesi arttıkça döngülerin sayısı dolayısıyla da işlem miktarı artacaktır. İki örnek alma arasında yapılan bu işlem miktarının artması örnekleme frekansının doğrudan düşmesine neden olmaktadır. Yani gerçekte filtre derecesine göre örnekleme süresi değişim göstermektedir. Buna karşın matlab ile yazılan sayısal filtre katsayılarının bulunduğu programda, kartın çalışma hızı 5kHz olarak alınmıştır. Küçük dereceli filtrelerde (1. ve 2. derecelerde) bu sorun olmazken 3. derereden itibaren örneklemenin yavaşlığı öngörülen sayısal filtre karakteristiğinden uzaklaşılmasına neden olmaktadır.

8 bitlik veri akışına karşın 12 bitlik örneklenmiş işaret bilgileri kullanılmıştır. İşaretli işlemlerin gerçekleştirilmesi için ADC'den gelen örnekler ve ara değer olarak kullanılan veriler üzerinde, filtre katsayılarıyla yapılan işlemlerde 16 bitlik işlemler yapılmıştır. Bilgisayarlarda kullanılan işaretli işlem mantığı benzer şekilde burada kullanılarak 12.nci bit üzerindeki bitler sıfır ise sayı pozitif veya hepsi bir ise sayı negatif olarak alınmıştır. Yani gerçek anlamda 12 bitlik sinyal işleme yapılmış ve işlemler de işaretli olarak düzenlenmiştir.

İşaretli işlemler bilgisayarlardaki gibi iki tümleyenini (two's complement) alma yöntemi gözönünde bulundurularak yapılmıştır. Yine bilgisayarlardaki gibi 16 bitlik sayılarla yapılan çarpımlarda oluşan 32 bitlik sonucun yüksek 16 biti alınarak düzenlenmesi (wordlength variation) yapılmıştır. Bu işlem Şekil 4.2 ile gösterilmiştir.



Şekil 4.1 Sayısal Filtre Akış Diagramı

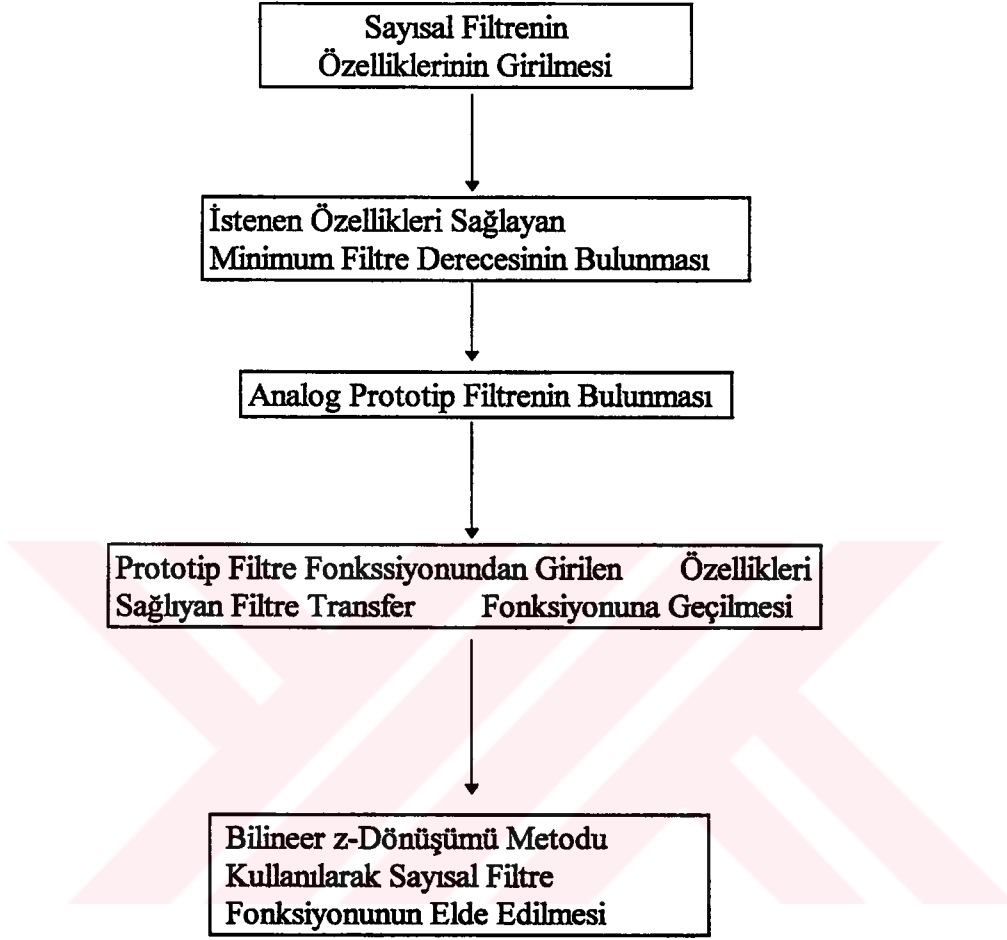


Şekil 4.2 İşaretli Çarpmada Bit Düzenleme

*X: Hex Sayı *S:4 Bitlik İşaret (0h veya Fh)

Dijital filtre katsayılarının bulunmasında matematik işlem kapasitesinin yüksekliğinden ve program yazılabilirliğinden dolayı matlab seçilmiştir. Yine matlab'teki özelleştirilmiş program grupları katsayıların bulunmasında büyük kolaylık sağlamıştır. Bu hazır fonksiyonlar istenen özellikleri sağlayan analog filtre fonksiyonlarını girilen parametrelere göre belirlemektedir. Sayısal filtre fonksiyonunun katsayılarının bulunduğu programın akış diagramı Şekil 4.3'te gösterilmiştir.

Program sonunda, elde edilen katsayılar Pascal programı tarafından okunup oradan da mikrokontrolöre atılmak üzere num.txt ve den.txt dosyalarına yazılmaktadır. Pasacalda yazılan programda ise dijital filtre katsayıları bu dosyalardan okunup dizi elemanlarına atanmaktadır. Ardından küçük olan veya olabilecek bu değerler daha sonra 80C32 filtre programında sonuç değer (filtrelenmiş bilgi) $n-2$ 'ye bölünmek üzere 2^{n-2} (n, ADC ve DAC'tan gelen sinyalin bit sayısı) ile çarpılmaktadır. Son olarakta bu değerler 8 bitlik paketler halinde (çünkü çarpma sonucu değerler 16 bitlik olarak kaydedildiler) sei haberleşme ile 80C32'ye aktarılmaktadır. Bu programlar Ek 1 ve Ek 2' de verilmiştir.



Şekil 4.3 Matlab İle Sayısal Filtre Katsayılarının Elde Edilmesi

5. SONUÇ

Gelişmiş DSP tümdevreleri yanında mikroişlemciler ve mikrokontrolörle dijital filtre tasarımı yapılmasının nedeni daha ucuz olmalarıdır. Bu çalışmada mikrokontrolörle dijital filtre tasarımı ele alınmış ve ucuzluğa karşı sınırlamaların neler olduğu incelenmiştir.

Çalışmada kullanılan 80C32 mikrokontrolörün ortalama komut süresinin 850ns ve veri yolunun 8 bit olması, örnekleme periyodunun artmasına bir başka deyişle örnekleme hızının azalmasına neden olmuştur. Buda düşük frekanslarda ve düşük dereceli transfer fonksiyonlarıyla çalışma zorunluluğu getirmiştir. 3. Dereceye kadar sonuç alınmıştır.

Analog işareti dijital, dijital işareti analoge dönüştürmek için kullanılan ADC774 ve DAC7801 tümdevreleriyle 12 bitlik bipolar örnekleme yapılmıştır. $\pm 10V$ arasında örneklenmiş değerlerin işlenmesi için iki baytlık işaretli aritmetik işlemleri yapan programlar yazılmıştır.

Daha yüksek hızlı ve 16 bit veya daha üstü işlem hacmine sahip mikroişlemci veya mikrokontrolörle söz konusu sınırlamalar doğal olarak daha azalacaktır.

KAYNAKLAR:

- 1) Arthur B. WILLIAMS, Fred J. TAYLOR; "ELECTRONIC FILTER DESIGN HANDBOOK"; McGraw-Hill Publishing Company; 1998
- 2) Andreas ANTONIOU; "DIGITAL FILTERS ANALYSIS, DESIGN and APPLICATION"; McGraw-Hill Inc.; 1993
- 3) C. Britton RORABAUGH; "DIGITAL FILTERS DESIGNERS HANDBOOK"; McGraw-Hill Inc.; 1993
- 4) H. HOLTZ; "THE ANALYSIS and SYNTHESIS of DIGITAL FILTERS"; 1975
- 5) T.W. PARKS, C.S. BURRUS; "DIGITAL FILTER DESIGN"; A Wiley-Interscience Publication; 1987
- 6) Leland B. JACKSON; "DIGITAL FILTERS & SIGNAL PROCESSING"; Kluwer Academic Publishers; 1996
- 7) Rodger E. ZIEMER, William H. TRANTER, D. Ronald FANNIN; "SIGNALS & SYSTEMS: CONTINUOUS and DISCRETE"; Macmillan Publishing Company; 1990
- 8) Lawrence R. RABINER, Bernard GOLD; "THEORY and APPLICATION of DIGITAL SIGNAL PROCESSING"; Prentice-Hall Inc.; 1975
- 9) A.V. OPPENHEIM; "DIGITAL SIGNAL PROCESSING"; Prentice-Hall Inc.; 1975
- 10) G. PROAKIS, G.M. MANOLAKIS; "INTRODUCTION to DIGITAL SIGNAL PROCESSING"
- 11) IEEE Micro Vol. 1. No.1 Feb. 1981; "DIGITAL FILTER IMPLEMENTATION on 16 BIT MICROCOMPUTERS"
- 12) D.M. ETTER; "ENGINEERING PROBLEM SOLVING with MATLAB"; Prentice-Hall, 1993
- 13) "TURBO PASCAL® USER'S GUIDE, Version 5.0", Borland International; 1988
- 14) "TURBO ASSEMBLER® Version 2.0, Quick Reference Guide"; 1990

EK 1: 80C32 İLE YAZILMIŞ DİJİTAL FİLTRE MAKİNA DİLİ PROGRAMI

;80C32 MİKROKONTROLÖR SAYISAL FİLTRE GERÇEKLEMESİ

```
; D(n-i) --> 30h-4fh
; Ai --> 70h-8fh
; Bi --> 90h-a0h
; us --> C0h
; x(n) --> C1h x(n)low - C2h x(n) high
; y(n) --> D8h y(n)low - D9h y(n) high
```

```
NAME Math_16_Module
;
Public Load_16,?Load_16?byte
Public Mul_16,?Mul_16?byte
Public Div_16,?Div_16?byte
Public Load_16,?Load_16?byte
Public Add_16,?Add_16?byte
Public Sub_16,?Sub_16?byte
;
Math_16_Data Segment Data
Math_16_Data Segment Code
;
RSEG Math_16_Data
```

```

?Load_16?byte : DS 2
?Mul_16?byte : DS 2
?Div_16?byte : DS 2
?Add_16?byte : DS 2
?Sub_16?byte : DS 2
OP_0:      DS 1
OP_1:      DS 1
OP_2:      DS 1
OP_3:      DS 1
TMP_0:     DS 1
TMP_1:     DS 1
TMP_2:     DS 1
TMP_3:     DS 1
DN_0 :     DS 1
DN_1 :     DS 1
;
RSEG  Math_16_Code

org  0000h
ljmp prog

```

```

;*****

```

;SERİ HABERLERLEŞME ALT PROGRAMI

```

prog:  clr ea                ;interruptlar aktif değil
      mov psw
      mov sp,#6eh
      mov pcon,#00h

```

```

mov tmod,#21h           ;timer1 8 bit reload timer
                        ;timer0 16 bit timer

mov tl1,#00h
mov th1,0fdh           ;9600 baudrate
mov tcon,#40h         ;timer1 başlat kontrol bit=1,
mov scon,#50h
mov a,sbuf             ;seri bufer başlangic durumu
mov ie,#80h           ;interruptlar aktif

s_inp: jnb ri,s_inp
      clr ri
      mov a,sbuf
      cjne a,#0fh,us   ;üs kod kontrol
      sjmp s_inp

us:   jnb ri,us
      clr ri
      mov a,sbuf
      mov c0h,a        ;us --> c0h

a_gir: jnb ri,a_gir
      clr ri
      mov a,sbuf
      cjne a,#66h,a_kats
      cjne a,#99h,b_kats
      sjmp a_gir

a_kats: mov a,c0h      ;Dijital Filtre payda polinomu katsayılarının alınması
      rl a             ;2*üs
      inc a            ;2*üs+1
      mov r2,a

```

```

        mov r0,#50h
bkl1:  jnb ri,bkl1
        clr ri
        mov a,sbuf          ;ai low ---->(50+i)h
        mov @r0,a          ;ai high ---->(50+i+1)h
        inc r0
        dec r2
        cjnz r2,#00h,bkl1
        ljmp a_gir

```

```

b_kats:mov a,c0h          ;Dijital Filtre pay polinomu katsayılarının alınması
        rl a              ;2*us
        inc a             ;2*us+1
        mov r2,a
        mov r0,#70h
bekl2: jnb ri,bekl2
        clr ri
        mov a,sbuf        ;bi low ---->(50+i)h
        mov @r0,a         ;bi high ---->(50+i+1)h
        inc r0
        dec r2
        cjnz r2,#00h,bekl2

```

;**ID DOĞRUDAN FİLTRE YAPISI İLE YAZILMIŞ DİJİTAL FİLTRE ALT
;PROGRAMI**

```

filt:  mov ie,#00h
        setb p1.7
        mov dptr,#2000h

```

```

    movx @dptr,a           ;adc'ye çevrimi başlat sinyali
qw:  jb  p3.2,qw
    lcall adc12           ;adc'den bilginin alınması
program: mov r0,#90h
    mov op_0,0c1h
    mov op_1,0c2h
    mov ?mul_16?byte+1,@r0
    inc r0
    mov ?mul_16?byte,@r0  ;İlk örnek için çıkışın hesaplanması
    call mul_16           ;y(n)=B(0)*D(n)
    mov a,tmp_1          ;Matlab'te katsayılar çarpıldığından dolayı sonuç
                        ; 210'a bölünüyor

```

```

rr a
rr a
mov r5,a
mov a,tmp_2
rrc a
rrc a
rrc a
anl a,#0c0h
orl a,r5
mov 0d8h,a
mov a,tmp_2
rr a
rr a
mov r5,a
mov a,tmp_3
rrc a
rrc a
rrc a

```

```

anl a,#0c0h
orl a,r5
mov 0d9h,a
mov dptr,#2000h
movx @dptr,a
lcall dac12
mov r0,#32h
mov @r0,0c1h
inc r0
mov @r0,0c2h
lcall adc12
tekrarla:mov dn_0,#00h
mov dn_1,#00h
mov dn_2,#00h
mov dn_3,#00h
mov r0,#72h ;Ai katsayılarının başlangıcı → R0
mov r1,#32h ;D(n-i) ara değerlerinin başlangıcı → R1
tekrar: mov op_0,@r1 ;D(n-i) low
inc r1
mov op_1,@r1 ;D(n-i) high
mov ?mul_16?byte+1,@r0 ;Ai low
inc r0
mov ?mul_16?byte,@r0 ;Ai high
call mul_16 ;Ai*D(n-i)
clr c
mov a,tmp_0 ;ΣAi*D(n-i) toplamının yapılması
add a,dn_0
mov dn_0,a
mov a,tmp_1
addc a,dn_1

```

```

mov dn_1,a
mov a,tmp_2
addc a,dn_2
mov dn_2,a
mov a,tmp_3
addc a,dn_3
mov dn_3,a
inc r0
inc r1
dec r2
cjne r2,#00h,tekrar

```

```

mov a,dn_1 ; $\Sigma A_i * D(n-i)$  toplamsonucunun  $2^{10}$ 'a bölünmesi
rr a ;bölüm işaretli olarak yapılıyor
rr a
mov r5,a
mov a,dn_2
rrc a
rrc a
rrc a
anl a,#0c0h
orl a,r5
mov dn_0,a
mov a,dn_2
rr a
rr a
mov r5,a
mov a,dn_3
rrc a
rrc a

```



```

rrc a
anl a,#0c0h
orl a,r5
mov dn_1,a
clr c                ;D(n)=X(i) - ΣAi*D(n-i) farkının hesaplanması
mov a,0c1h
subb a,dn_0
mov dn_0,a
mov a,0c2h
subb a,dn_1
mov dn_1,a
mov r0,#30h
mov @r0,dn_0
inc r0
mov @r0,dn_1
mov r2,0c0h
mov r0,#90h         ;Bi katsayılarının başlangıcı → R0
mov r1,#30h         ;D(n-i) ara değerlerinin başlangıcı →R1
mov dn_0,#00h
mov dn_1,#00h
tekr1: mov op_0,@r1   ;D(n-i) low
inc r1
mov op_1,@r1        ;D(n-i) high
mov ?mul_16?byte+1,@r0 ;Bi low
inc r0
mov ?mul_16?byte,@r0 ;Bi high
call mul_16         ;Bi*D(n-i)

clr c                ;y(n)=ΣBi*D(n-i) toplamının yapılması

```

```

mov a,tmp_0
add a,dn_0
mov dn_0,a
mov a,tmp_1
addc a,dn_1
mov dn_1,a
mov a,tmp_2
addc a,dn_2
mov dn_2,a
mov a,tmp_3
addc a,dn_3
mov dn_3,a
inc r0
inc r1
dec r2
cjne r2,#00h,tekr1

```

```

mov a,dn_1

```

;y(n)= $\sum B_i * D(n-i)$ toplam sonucunun 2^{10} 'a
; bölünmesi. Bölüm işaretli olarak yapılıyor.

```

rr a
rr a
mov r5,a
mov a,dn_2
rrc a
rrc a
rrc a
anl a,#0c0h
orl a,r5
mov dn_0,a
mov a,dn_2
rr a

```

```

rr a
mov r5,a
mov a,dn_3
rrc a
rrc a
rrc a
ani a,#0c0h
orl a,r5
mov dn_1,a
mov 0d8h,dn_0           ;y(n) low
mov 0d9h,dn_1           ;y(n) high
mov dptr,#2000h         ;Bir sonraki örnek için çevrimi başlat sinyali
movx @dptr,a
lcall dac12             ;Filtrelenmiş değerin DAC'a gönderilmesi
mov r2,0c0h             ;us → r2
mov a,r2
inc a
rl a                     ;2*us
add a,#30h              ;Geçici değerlerin bir sonraki filtreleme işlemi için
                        ; hafızada kaydırılmaları
mov r0,a                ;R0 ← D(n+inth)
redesig:dec r0
dec r0
mov a,@r0
inc r0
inc r0                   ;D(n-i-2) ---> D(n-i)
mov @r0,a
dec r0
dec r2
cjne r2,#02h,redesig

```

```
mov r2,0c0h
lcall adc12           ;ADC'den örneklenmiş değerin uygun formatta
                    ; alınması
lcall tekrarla
```

```
mul_16: mov tmp_2,#00h           ;2 byte'lık işaretli çarpma.
        mov tmp_3,#00h           ;Sonuç 4 byte
        mov b,op_0
        mov a,?mul_16?byte+1
        mul ab
        mov tmp_0,a
        mov tmp_1,b
        mov b,op_1
        mov a,?mul_16?byte+1
        mul ab
        clr c
        add a,tmp_1
        mov tmp_1,a
        jnc biriki
        inc b
```

```
biriki: mov tmp_2,b
        mov b,op_0
        mov a,?mul_16?byte
        mul ab
        clr c
        add a,tmp_1
        mov tmp_1,a
        mov a,b
        addc a,tmp_2
        mov tmp_2,a
```

```

jnc ikibir
mov a,tmp_3
inc a
mov tmp_3,a
ikibir: mov b,op_1
mov a,?mul_16?byte
mul ab
clr c
add a,tmp_2
mov tmp_2,a
mov a,b
addc a,tmp_3
mov tmp_3,a
ret

adc12: mov dptr,#2000h
movx a,@dptr ;ADC'den yüksek 8 bitin alınması
mov r3,a
inc dptr
movx a,@dptr ;ADC'den düşük 4 bitin alınması
anl a,#0f0h ;Örneğin SX XX formatına çevrilip hafızaya
mov r4,a ;alınması
mov a,r4
rr a
rr a
rr a
rr a
mov r4,a
mov a,r3
anl a,#0fh

```

```

rl a
rl a
rl a
rl a
ori a,r4
mov 0c1h,a
mov a,r3
andi a,#0f0h
rr a
rr a
rr a
rr a
clr c
subb a,#08h
mov 0c2h,a
ret

```

```

dac12: mov dptr,#4000h      ;SX XX formatındaki sonuç değerini
mov a,0d8h                ;DAC'a gönderilmesi
movx @dptr,a              ;y(n) low ( düşük 8 bit ) → DAC
inc dptr
mov a,0d9h                ;y(n) high ( yüksek 4 bit ) → DAC
add a,#08h
movx @dptr,a
mov dptr,#6000h
movx @dptr,a
ret

```

EK2: MATLAB İLE DİJİTAL FİLTRE KATSAYILARININ BULUNMASI

% BİLİNEER z-TRANSFORMU YÖNTEMİ İLE DİJİTAL FİLTRE KATSAYILARI
% BULUNMAKTADIR

```
fs=3e3;
```

```
fn=fs/2;
```

% 80C32 Mikrokontrolörünün yapısından ve 80C32'de yazılan programdan dolayı
%örnekleme frekansı 3kHz olarak alınmıştır.

```
tip=menu('IIR Filtre Tipini Giriniz','Butterworth','Chebyshev','Elliptic');
```

```
if tip==1
```

```
    ftype=menu('Buterworth Filtre Tipini  
Giriniz','LowPass','HighPass','BandPass','BandStop');
```

```
    if ftype==3
```

```
        wp1=input('Geçirme Bandı Alt Kesim Frekansı (Hz) Wp1=');
```

```
        wp2=input('Geçirme Bandı Üst Kesim Frekansı (Hz) Wp2=');
```

```
        wp1=wp1/fn;
```

```
        wp2=wp2/fn;
```

```
        wp=[wp1 wp2];
```

```
ws1=input('Söndürme Bandı Alt Frekansı (Hz) Ws1=');  
ws2=input('Söndürme Bandı Üst Frekansı (Hz) Ws2=');
```

```
ws1=ws1/fn;  
ws2=ws2/fn;
```

```
ws=[ws1 ws2];
```

```
rp=input('Geçirme Bandı Dalgalanması (dB) Rp=');  
rs=input('Söndürme Bandı Zayıflatması (dB) Rs=');
```

```
elseif ftype==4
```

```
wp1=input('Geçirme Bandı Alt Frekansı (Hz) Wp1=');  
wp2=input('Geçirme Bandı Üst Frekansı (Hz) Wp2=');
```

```
wp1=wp1/fn;  
wp2=wp2/fn;
```

```
wp=[wp1 wp2];
```

```
ws1=input('Söndürme Bandı Alt Frkansı Ws1=');  
ws2=input('Söndürme Bandı Üst Frekansı Ws2=');
```

```
ws1=ws1/fn;  
ws2=ws2/fn;
```

```
ws=[ws1 ws2];
```

```
rp=input('Geçirme Bandı Dalgalanmsı (dB) Rp=');
```



```

rs=input('Söndürme Bandı Zayıflatması (dB) Rs=');

else
    wp=input('Geçirme Bandı Kesim Frekansını Giriniz (Hz) Wp=');
    ws=input('Söndürme Bandı Kesim Frekansını Giriniz (Hz) Ws=');

wp=wp/fn;
ws=ws/fn;

rp=input('Geçirme Bandı Dalgalanması (dB) Rp=');
rs=input('Söndürme Bandı Zayıflatması (dB) Rs=');
end;

[n, wn]=buttord(wp, ws, rp, rs);

if ftype==1
    [num, den]=butter(n, wn);

elseif ftype==2
    [num, den]=butter(n, wn, 'high');

elseif ftype==3
    [num, den]=butter(n, wn);

elseif ftype==4
    [num, den]=butter(n, wn, 'stop');

end;

elseif tip==2

```

```
ftype=menu('Chebyshev Filtre Türünü  
Giriniz','LowPass','HighPass','BandPass','BandStop');
```

```
if ftype==3
```

```
wp1=input('Geçirme Bandı Alt Kesim Frekansı (Hz) Wp1=');
```

```
wp2=input('Geçirme Bandı Üst Kesim Frekansı (Hz) Wp2=');
```

```
wp1=wp1/fn;
```

```
wp2=wp2/fn;
```

```
wp=[wp1 wp2];
```

```
ws1=input('Söndürme Bandı Alt Frekansı (Hz) Ws1=');
```

```
ws2=input('Söndürme Bandı Üst Frekansı (Hz) Ws2=');
```

```
ws1=ws1/fn;
```

```
ws2=ws2/fn;
```

```
ws=[ws1 ws2];
```

```
rp=input('Geçirme Bandı Dalgalanması (dB) Rp=');
```

```
rs=input('Söndürme Bandı Zayıflatması (dB) Rs=');
```

```
elseif ftype==4
```

```
wp1=input('Geçirme Bandı Alt Frekansı (Hz) Wp1=');
```

```
wp2=input('Geçirme Bandı Üst Frekansı (Hz) Wp2=');
```

```
wp1=wp1/fn;
```

```
wp2=wp2/fn;
```

```
wp=[wp1 wp2];
```

```
ws1=input('Söndürme Bandı Alt Frekansı (Hz) Ws1=');
```

```
ws2=input('Söndürme Bandı Üst Frekansı (Hz) Ws2=');
```

```
ws1=ws1/fn;
```

```
ws2=ws2/fn;
```

```
ws=[ws1 ws2];
```

```
rp=input('Geçirme Bandı Dalgalanması (dB) Rp=');
```

```
rs=input('Söndürme Bandı Zayıflatması (dB) Rs=');
```

```
else wp=input('Geçirme Bandı Kesim Frekansı (Hz) Wp=');
```

```
ws=input('Söndürme Bandı Kesim Frekansı (Hz) Ws=');
```

```
wp=wp/fn;
```

```
ws=ws/fn;
```

```
rp=input('Geçirme Bandı Dalgalanması (dB) Rp=');
```

```
rs=input('Söndürme Bandı Zayıflatması (dB) Rs=');
```

```
end;
```

```
[n, wn]=cheblord(wp, ws, rp, rs);
```

```
if ftype==1
```

```
    [num, den]=cheby1(n, rp, wn);
```

```
elseif ftype==2
```

```

[num, den]=cheby1(n, rp, wn, 'high');

elseif ftype==3
    [num, den]=cheby1(n, rp, wn);

elseif ftype==4
    [num, den]=cheby1(n, rp, wn, 'stop');

end;

elseif tip==3
    ftype=menu('Elliptic Filtre Türünü
Giriniz','LowPass','HighPass','BandPass','BandStop');

    if ftype==3
        wp1=input('Geçirme Bandı Alt Kesim Frekansı (Hz) Wp1=');
        wp2=input('Geçirme Bandı Üst Kesim Frekansı (Hz) Wp2=');

        wp1=wp1/fn;
        wp2=wp2/fn;

        wp=[wp1 wp2];

        ws1=input('Söndürme Bandı Alt Frekansı (Hz) Ws1=');
        ws2=input('Söndürme Bandı Üst Frekansı (Hz) Ws2=');

        ws1=ws1/fn;
        ws2=ws2/fn;

        ws=[ws1 ws2];

```

```

rp=input('Geçirme Bandı Dalgalanması (dB) Rp=');
rs=input('Söndürme Bandı Zayıflatması (dB) Rs=');

elseif ftype==4
wp1=input('Geçirme Bandı Alt Frekansı (Hz) Wp1=');
wp2=input('Geçirme Bandı Üst Frekansı (Hz) Wp2=');

wp1=wp1/fn;
wp2=wp2/fn;

wp=[wp1 wp2];

ws1=input('Söndürme Bandı Alt Frekansı (Hz) Ws1=');
ws2=input('Söndürme Bandı Üst Frekansı (Hz) Ws2=');

ws1=ws1/fn;
ws2=ws2/fn;

ws=[ws1 ws2];

rp=input('Geçirme Bandı Dalgalanması (dB) Rp=');
rs=input('Söndürme Bandı Zayıflatması (dB) Rs=');

else wp=input('Geçirme Bandı Kesim Frekansı (Hz) Wp=');
ws=input('Söndürme Bandı Kesim Frekansı (Hz) Ws=');

wp=wp/fn;
ws=ws/fn;

```

```
rp=input('Geçirme Bandı Dalgalanması (dB) Rp=');
rs=input('Söndürme Bandı Zayıflatması (dB Rs=');
```

```
end;
```

```
[n, wn]=ellipord(wp, ws, rp, rs);
```

```
if ftype==1
```

```
    [num, den]=ellip(n, rp, rs, wn);
```

```
elseif ftype==2
```

```
    [num, den]=ellip(n, rp, rs, wn, 'high');
```

```
elseif ftype==3
```

```
    [num, den]=ellip(n, rp, rs, wn);
```

```
elseif ftype==4
```

```
    [num, den]=ellip(n, rp, rs, wn, 'stop');
```

```
end;
```

```
end;
```

```
if ftype <= 2 us=n+1;
```

```
    else us=2*n+1;
```

```
end;
```

```
[h, wt]=freqz(num, den, 1000);
```

```
t=10e-6;
```

```
hertz=wt/(2*pi*t);  
gen=abs(h);  
hz=hertz;  
plot(hz, gen); title('Magnitude');  
xlabel ('Frequency(Hz)'), ylabel('Magnitude'),grid;
```

```
[fid,message] = fopen(' num.txt ','w');  
fprintf(fid,'%2.4f\n',num);  
fclose(fid);
```

```
[fid,message] = fopen(' den.txt ','w');  
fprintf(fid,'%2.4f\n',den);  
fclose(fid);
```

```
[fid,message] = fopen(' derece.txt ','w');  
fprintf(fid,'%2.0f\n',us);  
fclose(fid);  
end
```

EK 3. PASCAL İLE MİKROKONROLÖRE DİJİTAL FİLTRE KATSAYILARININ AKTARILMASI

Program katsayı_gönder;

Uses Crt;

type

Dizi=Array[1..32] Of Longint;

Const

RSBA=\$03f8;

Var

Gdata, Adata :Byte;

Num_f : Text; Den_f :Text; derece_f :Text; katsayı :dizi;

k,l,j :Integer;

Procedure Hazırla;

Var

ByteBuf :Byte;

Begin

Port [RSBA+3] :=\$80; { open divisor latches}

Port [RSBA+1] :=0; { interrupt control }

Port [RSBA] :=\$0C; { baud rate =9600 Bps }

Port [RSBA+3] :=\$0F; { line control word }

ByteBuf := Port [RSBA+2]; { clear interrupt flag}

ByteBuf := Port [RSBA]; { clear receive buffer }

End;


```
Procedure AL ( Var OKUDATA : Bbyte );
```

```
  Var
```

```
  HataCod : Byte ;
```

```
  Begin
```

```
  Repeat
```

```
  HataCod := Port [ RSBA+5];
```

```
  Until ( HataCod and 1 ) =1;
```

```
  OKUDATA := Port [ RSBA];
```

```
  End;
```

```
Procedure Gonder( Var YAZDATA :Byte );
```

```
  Var
```

```
  HataCod :Byte;
```

```
  Begin
```

```
  Repeat
```

```
  HataCod := Port [ RSBA+5];
```

```
  until ( HataCod ans $40 ) > 0;
```

```
  Port [ RSBA];
```

```
  End;
```

```
Procedure num_oku;
```

```
  begin
```

```
  Assign(num_f, 'c:\matlab\bin\num.txt');
```

```
  Reset(num_f);
```

```
  i:=0;
```

```
  Repeat
```

```
  i:=i+1;
```

```
  Read(num_f,katsayı[i]);
```

```
Until eof(num_f);  
Close(num_f);  
End;
```

```
Procedure Den_oku;
```

```
begin  
Assign(den_f, 'c:\matlab\bin\den.txt');  
Reset(den_f);  
i:=0;  
Repeat  
i:=i+1;  
Read(den_f,katsayi[i]);  
Until eof(den_f);  
Close(den_f);
```

```
End;
```

```
Procedure Derece_oku;
```

```
begin  
Assign(derece_f, 'c:\matlab\bin\derece.txt');  
Reset(derece_f);  
Read(derece_f,k);  
Close(derece_f);
```

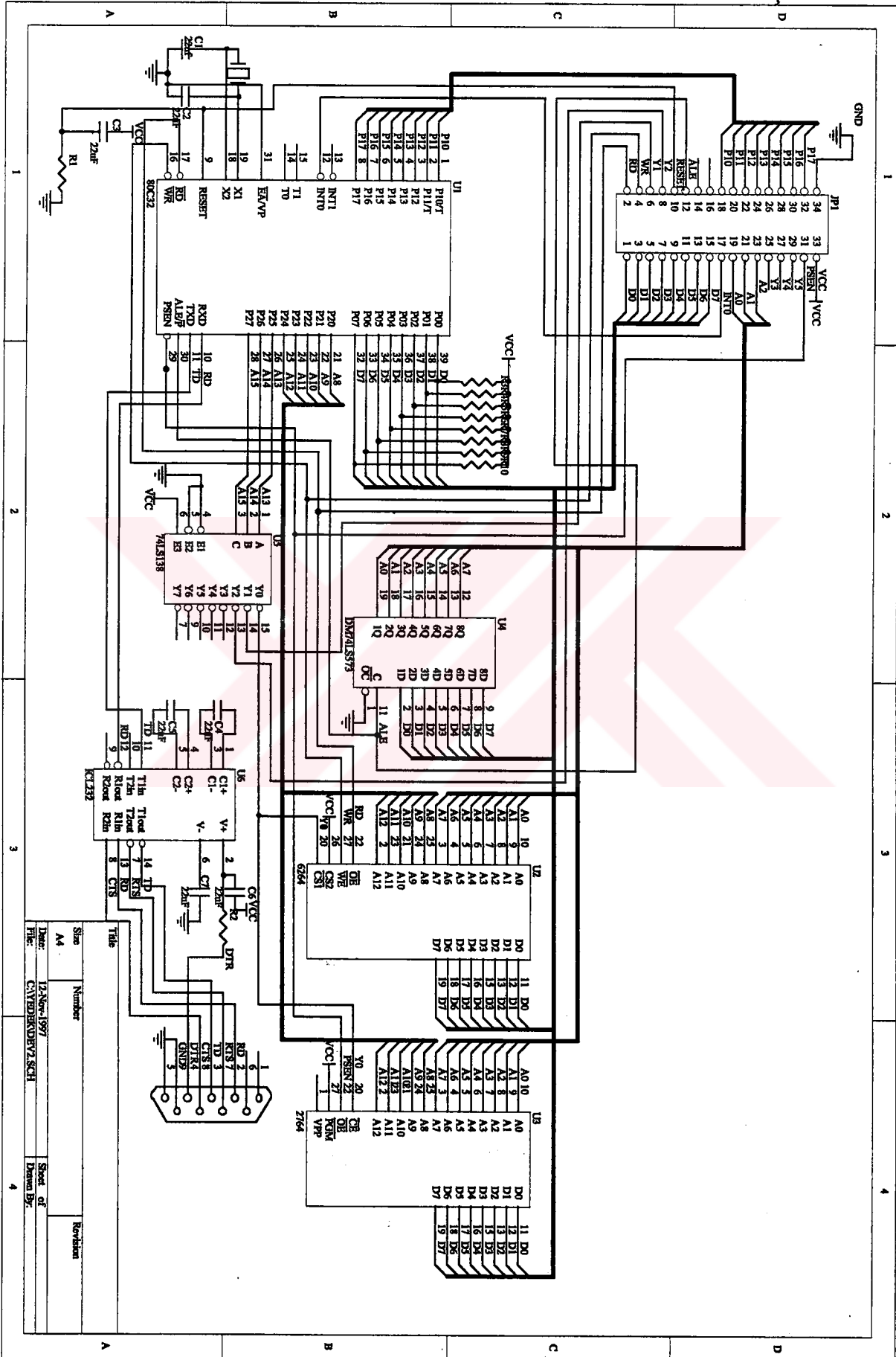
```
End;
```

```
BEGIN
```

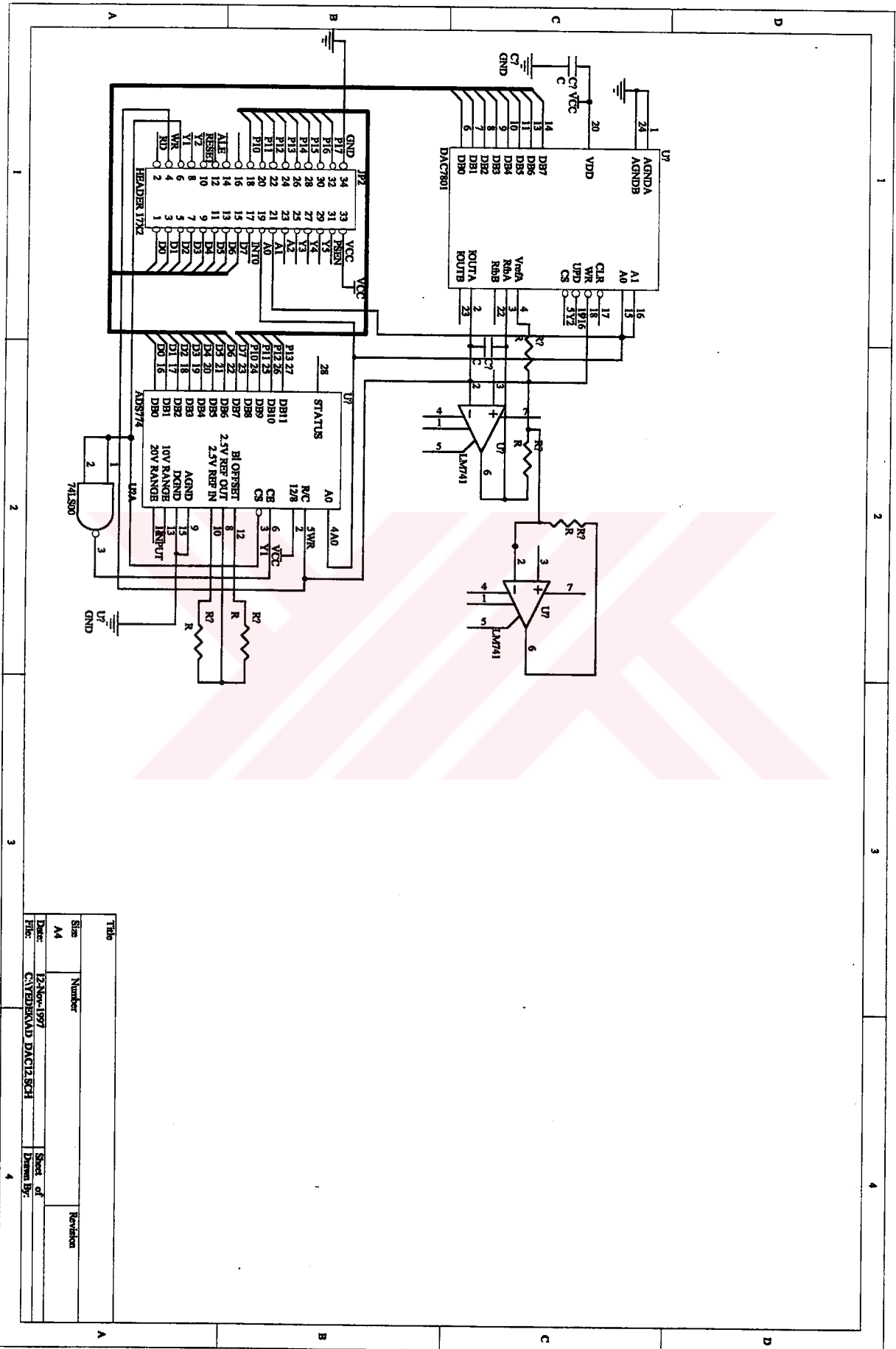
```
ClrScr;  
Hazirla;  
Derece_oku;  
Gdata:=$000f;  
Gonder(Gdata);
```

```
Gdata:=k;
Gonder(Gdata);
Den_oku;
Gdata:=$0066;
Gonder(Gdata);
For j:=1 To k Do
Begin
    Gdata:= katsayı [j] ;
    Gonder(Gdata);
End;
Gdata:=$00ff;
Gönder( Gdata );
Num_oku;
For j:=1 To k Do
Begin
    Gdata:= katsayı [j] ;
    Gonder(Gdata);
End;
End.
```

EK 4 80C32 MİKROKONTROLÖR KARTI SCHEMATIC ÇİZİMLERİ



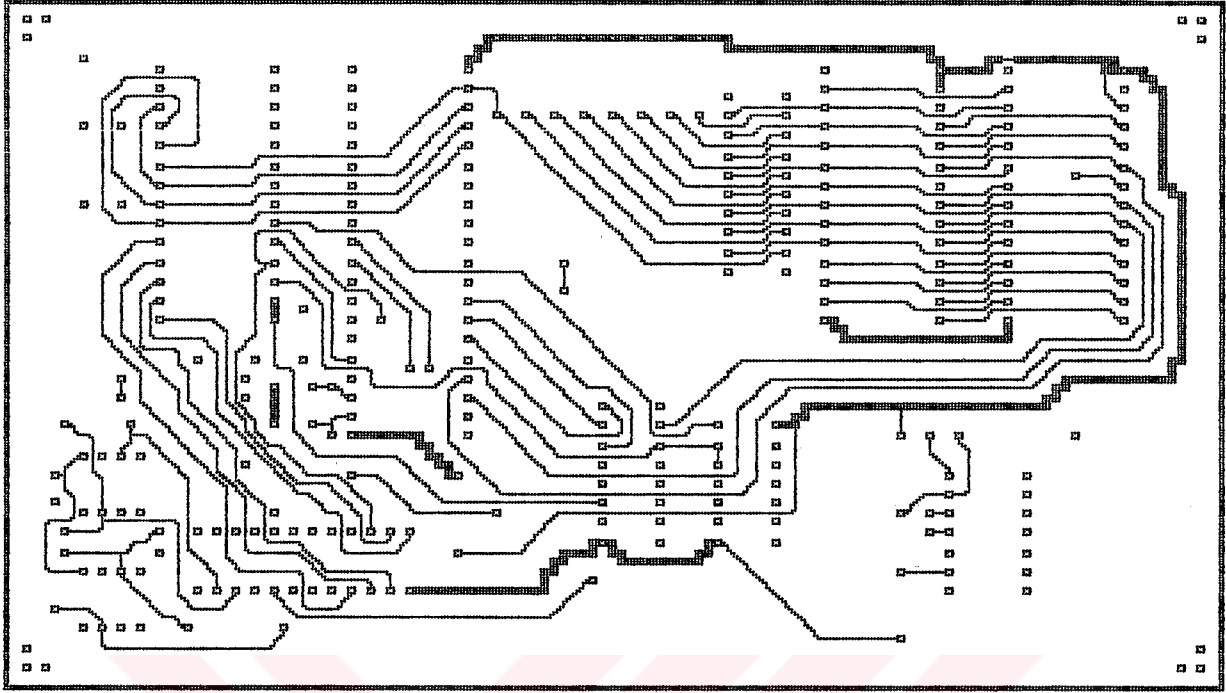
Size	Number	Revision
A4	12-Nov-1997	
Date:	12-Nov-1997	Sheet of
File:	C:\VBK\KOV\DVZ\SCH	Drawn By:



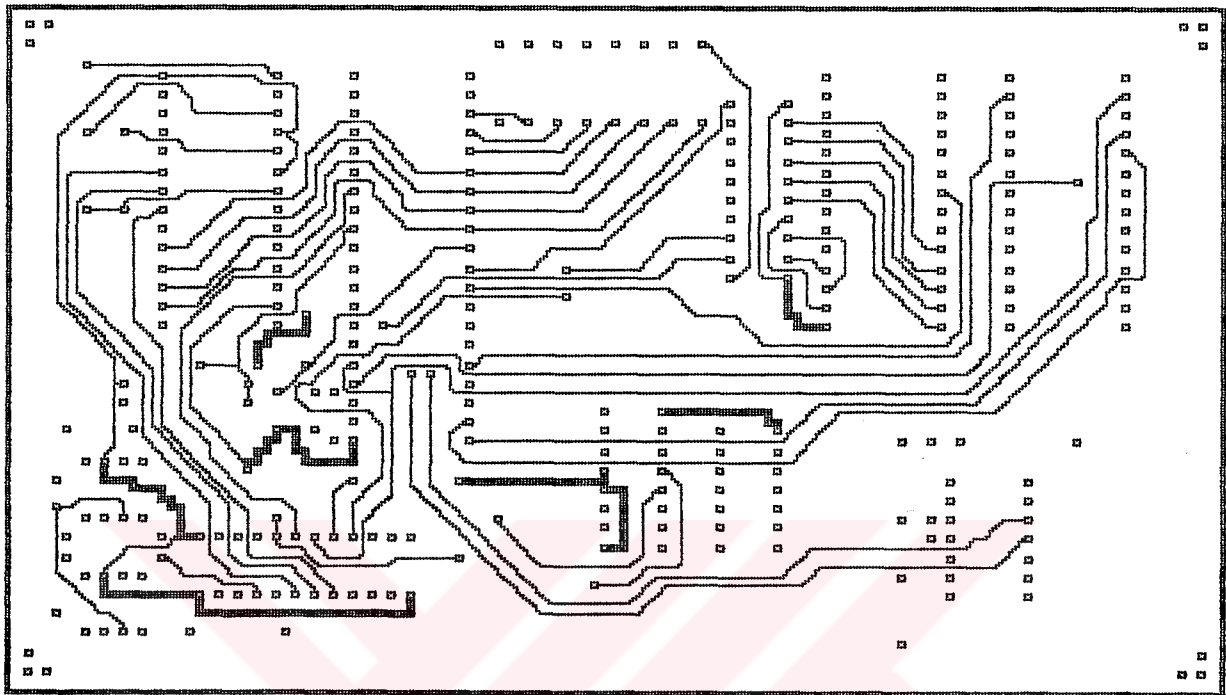
Title		Revision	
Size	Number		
A4			
Date:	12-Nov-1997	Sheet of	
File:	C:\VEDIKAWAD\DAC7801	Drawn By:	

EK 5 80C32 MİKROKONTROLÖR KARTI PCB ÇİZİMLERİ

1x checkplot 28 Oct 97 15:42:20
filtre
v1.4 r0 holes: 308 solder side
approximate size: 6.15 by 3.40 inches



1X checkplot 28 Oct 97 16:44:42
filtre
v1.4 r0 holes: 308 component side
approximate size: 6.15 by 3.40 inches



ÖZGEÇMİŞ

Doğum Tarihi : 14 Aralık 1972

Doğum Yeri : Amasya

Eğitim :

- 1987-1990 Çerkezköy Lisesi
- 1990-1994 Anadolu Üniv. Elek-Elektronik Fak.
- 1994-1997 Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü

