1. GİRİŞ

Analog devrelerden sayısal devrelere geçildiği günümüzde, sayısal işaret işlemede, sayısal filtreler önemli bir rol oynamaktadır. Birçok cihazda kullanılan sayısal filtreleri gerçeklemek için çeşitli yollar bulunmaktadır.

Bu çalışmada, sayısal bir sonsuz impuls yanıtlı (Infinite Impuls Response, IIR) filtre sahada programlanabilir kapı dizileri (Field Programmable Gate Arrays, FPGA) üzerinde tasarlanmıştır. Maliyetinin düşüklüğü, tasarlanan devreyi gerçekleme ve test etme süresinin kısalığı ve tekrar tekrar programlanabilmesinin getirdiği avantajlar dolayısıyla tasarımda FPGA kullanımı tercih edilmiştir.

Filtre tasarlanırken kayan nokta aritmetiği kullanılmıştır. Bu sisteme sonuçların kesinliği ve gerçeğe yakınlığı açısından fayda sağlamıştır.

Filtre sistemlerinin temel birimleri olan toplama ve çarpma devreleri parametrik olarak tasarlanmıştır. Sentez aşamasından önce belirli değişkenlere değer vererek devrelerin istenilen düzendeki kayan noktalı sayılarla işlem yapması sağlanabilmektedir.

Filtrenin gerçekleme aşamasında fark denklemindeki çarpma, toplama ve çıkarma işlemlerini gerçekleştirmek için direkt-I yapısı kullanılmıştır. Toplama ve çarpma devreleri gibi filtre de parametrik olarak tasarlanmıştır. Belirli değişkenlere değer verilerek filtrenin derecesi ve işlem yapacağı kayan noktalı sayıların düzeni belirlenebilmektedir.

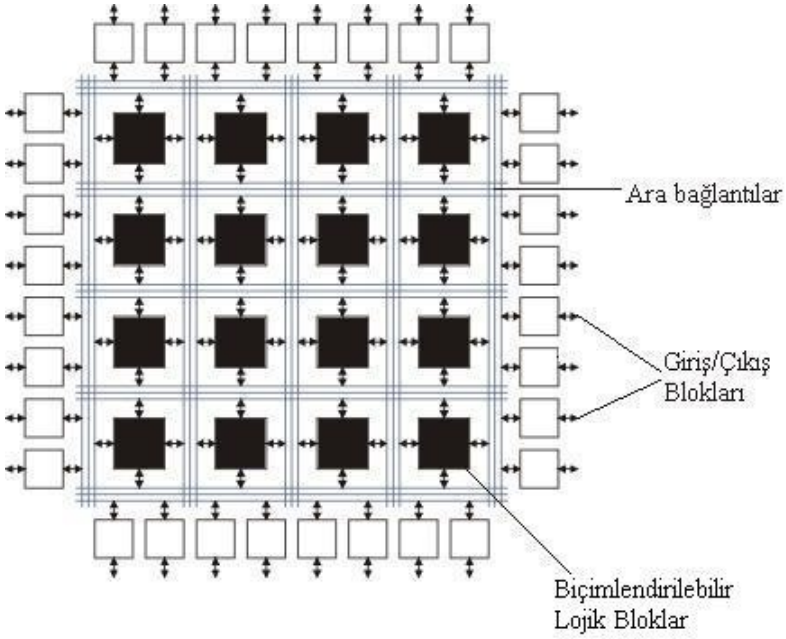
Tasarımın doğru bir şekilde çalışıp çalışmadığı ise üçüncü dereceden gerçeklenen bir filtrenin benzetimi yoluyla sınanmıştır.

2. SAHADA PROGRAMLANABİLİR KAPI DİZİLERİ (FPGA)

2.1 Genel FPGA Mimarisi

FPGA'lar sahada programlanabilen lojik (FPL) cihazlar ailesinin bir üyesidir. FPL'ler birbirini tekrar eden küçük lojik bloklar içeren, programlanabilir cihazlar olarak tanımlanır [1]. FPGA'lar da programlanabilir lojik bloklar ve ara bağlantılardan oluşmuştur. Bu lojik bloklar ve ara bağlantıların kullanıcı tarafından programlanmasıyla belirli bir fonksiyonu gerçekleyen devreler oluşturulur. Bu fonksiyonlar en temel lojik işlemler olabileceği gibi filtreleme gibi karmaşık işlemler de olabilir.

Bir FPGA'yı oluşturan temel yapılar Şekil 2.1'de görüldüğü gibi, biçimlendirilebilir lojik bloklar (Configurable Logic Blocks, CLB), giriş/çıkış blokları (Input/Output Blocks, IOB) ve ara bağlantılardır.



Şekil 2.1: FPGA mimarisi

Biçimlendirilebilir lojik bloklar genel olarak doğruluk tabloları (Look-up Table, LUT) ve flip-flop'lardan oluşmuştur. CLB'ler ara bağlantılarla birbirlerine bağlanabilir ve karmaşık fonksiyonlar gerçeklenebilir. IOB'ler FPGA içindeki sinyallerin dış ortamla olan bağlantısını sağlar. Ara bağlantılar ise uygun şekilde programlanarak CLB'ler ve IOB'leri birbirine bağlar [2].

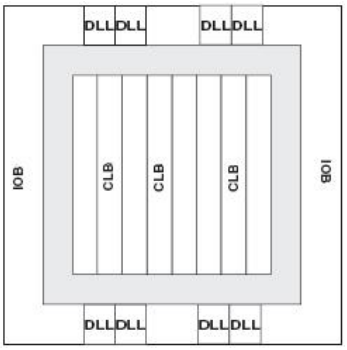
2.2 FPGA'ların Programlanması

FPGA'yı programlamak için gerekli ilk adım donanım tanımlama dilleri ile (Hardware Description Language, HDL) veya şematik yardımıyla, FPGA'nın davranışını tanımlamaktır. Daha sonra üretici tarafından sağlanan yazılım ile devreye ait bağlantı listesi (netlist) oluşturulur. Bu aşamada devrenin doğru bir şekilde çalışıp çalışmadığı bilgisayar ortamında benzetim (simulation) yapılarak test edildikten sonra varsa gerekli değişiklikler yapılır ve lojik sentezleme aşamasına geçilir. Lojik sentezleme esnasında devre fonksiyonları indirgenerek FPGA içindeki CLB'lerle eşleştirilir ve kapı seviyesinde bir bağlantı listesi oluşturulur. Bir sonraki aşama olan yerleştirme ve yollandırma (place and route) aşamasında da devre fonksiyonları ile eşleştirilmiş CLB'ler, FPGA içerisinde uygun yerlere yerleştirilir ve bu bloklar arasındaki bağlantılar oluşturulur. Gerçeğe çok yakın olan bu modelin benzetim yoluyla çalışması sınandıktan sonra FPGA üreticisi tarafından sağlanan yazılım ile FPGA'yı programlamak için gerekli kod oluşturulur. Son olarak uygun donanımlar yardımıyla FPGA bir önceki aşamada oluşturulan kodlarla programlanır ve işlem tamamlanır [2].

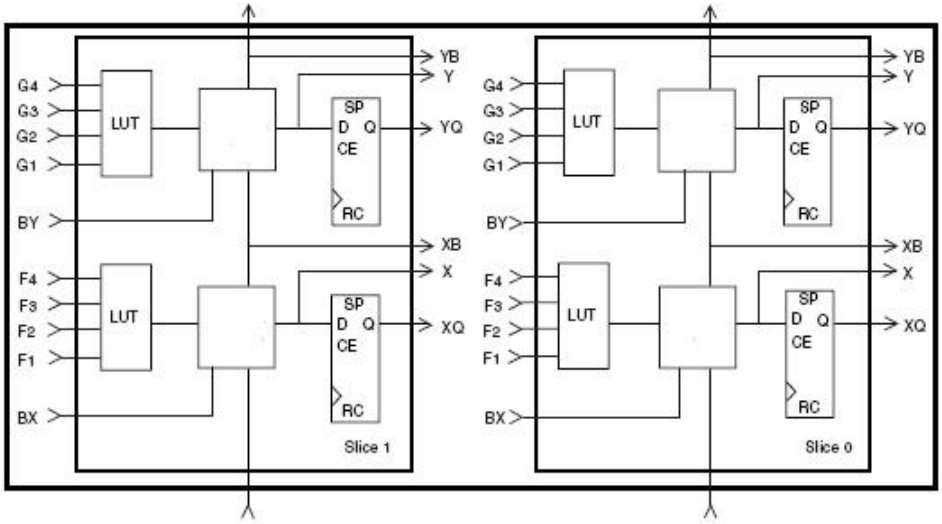
2.3 Tasarımda Kullanılan FPGA'nın Yapısı

Tasarımda Xilinx'in Virtex-E serisi FPGA'larından 1000E modeli kullanılmıştır. Bu modelin mimarisi Şekil 2.2'de görülmektedir.

CLB'lerin en temel yapısı lojik hücrelerdir. Her bir lojik hücre, dört girişli bir doğruluk tablosu, LUT tarafından sürülen bir blok ve bir saklama elemanından oluşur. Şekil 2.3'de görüldüğü gibi her CLB içerisinde birbirinin aynı iki dilim halinde düzenlenmiş toplam dört lojik hücre bulunur. Tablo 2.1'de de Virtex-E 1000E modelinin içerdiği toplam eleman sayıları görülmektedir [3].



Şekil 2.2: Virtex-E mimarisi



Şekil 2.3: İki dilimli CLB yapısı

Tablo 2.1: Virtex-E 1000E modelinin eleman dağılımı

Lojik Birimler	**Mevcut Sayı**
Dilimler	12288
Dilimlerdeki Flip-Floplar	24576
Dört Girişli LUT'lar	24576
IOB'ler	158
Saat	4

3. KAYAN NOKTA ARİTMETİĞİ

3.1 Kayan Noktalı Gösterim

Kayan noktalı gösterim, sabit noktalı gösterime göre çok büyük sayılarla çalışmak için daha geniş bir dinamik erim ve çok küçük sayılarda daha kesin sonuçlar sağlar.

Bu gösterilimdeki sayılar dört bölümden oluşur: işaret, üs *e*, üs tabanı *b* ve anlamlı kısım *s*. Bu bölümler bir sayıyı (3.1) eşitliğinde olduğu gibi temsil eder.

$$x = \pm s \times b^{e} \quad (3.1)$$

Burada dikkat edilmesi gereken kayan noktalı gösterimde iki tane işaretin olduğudur. Bunlardan biri sayının işareti, diğeri ise üssün işaretidir. Üssün işareti ile uğraşmamak için genelde yanlı (biased) gösterim kullanılır. Böylece üs belirli bir sayıyla toplanarak, üssün negatif olması engellenir. Şekil 3.1'de kayan noktalı gösterimin en genel hali görülmektedir.

±	***E***	**s**
İşaret 0: + 1: -	Üs İşaretli tamsayı, genelde yanlı gösterimle işaretsiz olarak kullanılır. n bit için aralık: [-yanlılık, 2^n-1-yanlılık] Yanlılık = 2^{n-1}-1	Anlamlı kısım Sabit nokta gösterimindedir.

Şekil 3.1: Kayan nokta gösterimi

Kayan nokta gösteriminde kullanılan standart "ANSI/IEEE Std 754-1985" standardıdır. Bu standartta üs yanlı biçimde ifade edilirken, anlamlı kısım sabit bir '1' ile başlar ve bu bite saklı bit denir. Anlamlı kısım *s, 1.f* şeklinde gösterilir ve *f*'nin bir diğer ismi de mantistir. Tek duyarlı gösterim biçimi 32 bit uzunluğunda, çift duyarlı gösterim biçimi ise 64 bit uzunluğundadır. Bu gösterimdeki kayan noktalı sayıların on tabanında temsil ettiği değer (3.1) ifadesine benzer olarak şu şekildedir:

$$x = (-1)^{isaret} \times 2^{e-yanlilik} \times 1.f \quad (3.2)$$

Kayan noktalı gösterimde 0 temsil edilemeyeceğinden dolayı işaret biti hariç bütün bitlerin sıfır olduğu durum 0 olarak kabul edilmiştir. Tablo 3.1'de IEEE standardındaki kayan noktalı sayıların bazı önemli özellikleri verilmiştir [4].

Tablo 3.1: IEEE standardı

Özellik	**Tek Duyarlı**	**Çift Duyarlı**

Toplam bit sayısı	32	64
Anlamlı bitler (*s*)	23 + 1 saklı bit	52 + 1 saklı bit
Üs bit sayısı	8	11
Üs yanlılığı	127	1023
En küçük değer	1.2×10^{-38}	2.2×10^{-308}
En büyük değer	3.4×10^{38}	1.8×10^{308}

3.2 Kayan Noktalı Gösterimde Toplama

Kayan noktalı sayılarda toplama yapmak sabit noktalı sayılarda toplama yapmaktan çok farklıdır. Toplama[1] tek aşamada değil aşağıda belirtildiği gibi dört aşamada gerçekleştirilir:

1. Üsler karşılaştırılarak aralarındaki fark bulunur.
2. Eğer üsler farklı ise küçük üsse ait olan mantis, fark kadar sağa kaydırılır. Böylece toplanacak iki sayının üsleri eşitlenmiş olur.
3. Kaydırılmış mantisler aynı işaretliyse toplanır, farklı işaretli ise çıkarılır. Sonuç işareti belirlenir.
4. Sonuç mantisi normalize edilerek 1.*f* standardına getirilir. Eğer gerekiyorsa üs de arttırılır veya azaltılır [5].

[1]Toplama esnasında saklı bit gelen sayıların bitlerine dahil değildir. İşlem sırasında mantise eklenir.

Şekil 3.2'de toplama işleminin akış diyagramı görülmektedir. İşlemin anlaşılması açısından Örnek 3.1'in incelenmesi faydalı olacaktır. İşlem kolaylığı açısından sayılar 8 bitlik (1 işaret, 4 üs, 3 mantis biti) alınmıştır.

Örnek 3.1: Kayan noktalı sayıların toplanması

$A = 0\ 1000\ 000$ $B = 0\ 1001\ 001$ Yanlılık = $2^{4-1}-1 = 7$

$A = 1.000\times2^{8-7} = 2$
$B = 1.001\times2^{9-7} = 4.5$ *A* ve *B*'nin onluk tabandaki değerleri.

$A' = 0.100\times2^{2}$
$B = 1.001\times2^{2}$ Üsler arasındaki fark 1, *A*'nın mantisi sağa 1 defa kaydırıldı.

$T = 1.101\times2^{2} = 6.5$ Sonuç mantisi 1.*f* standardında, normalize etmeye gerek yok.

Dolayısıyla üssü arttırmaya ya da azaltmaya da gerek yok.

$T = 0\ 1001\ 101$ İki sayıda pozitif olduğundan sonuç pozitif.

3.3 Kayan Noktalı Gösterimde Çarpma

Kayan noktalı sayılarda çarpma, toplamada olduğu gibi birden fazla aşamada gerçekleştirilir. Çarpma[1] aşamaları şu şekildedir:

1. Üsler toplanır ve toplamdan yanlılık çıkarılır.
2. Mantisler çarpılır.
3. Mantis çarpımı normalize edilerek sonuç mantisi 1.*f* standardına getirilir. Sonuç işareti belirlenir.
4. Gerekiyorsa üs arttırılır ya da azaltılır [6].

Şekil 3.3'de çarpma işleminin akış diyagramı görülmektedir. İşlemin anlaşılması açısından Örnek 3.2'in incelenmesi faydalı olacaktır. İşlem kolaylığı açısından sayılar 8 bitlik (1 işaret, 4 üs, 3 mantis biti) alınmıştır.

Örnek 3.2: Kayan noktalı sayıların çarpılması

$A = 0\ 1000\ 000$ $B = 1\ 1001\ 001$ Yanlılık = $2^{4-1}-1 = 7$

$A = 1.000 \times 2^{8-7} = 2$

$B = 1.001 \times 2^{9-7} = -4.5$ Sonuç Üssü = 8+9-7 = 10

[1]Çarpma esnasında saklı bit gelen sayıların bitlerine dahil değildir. İşlem sırasında mantise eklenir.

				1	0	0	0	
			×	1	0	0	1	
				1	0	0	0	
			0	0	0	0		
		0	0	0	0			
+	1	0	0	0				
	1	0	0	1	0	0	0	

A ve *B* mantislerinin çarpımı 1.*f* standardına çekildiğinde 1.001 olur.

$Ç = 1.001 \times 2^{10-7} = -9$ Mantis no yapmaya gerek yok.

$Ç = 1\ 1010\ 001$ Çarpım sonucu negatif.

4. TOPLAMA VE ÇARPMA DEVRELERİNİN GERÇEKLENMESİ

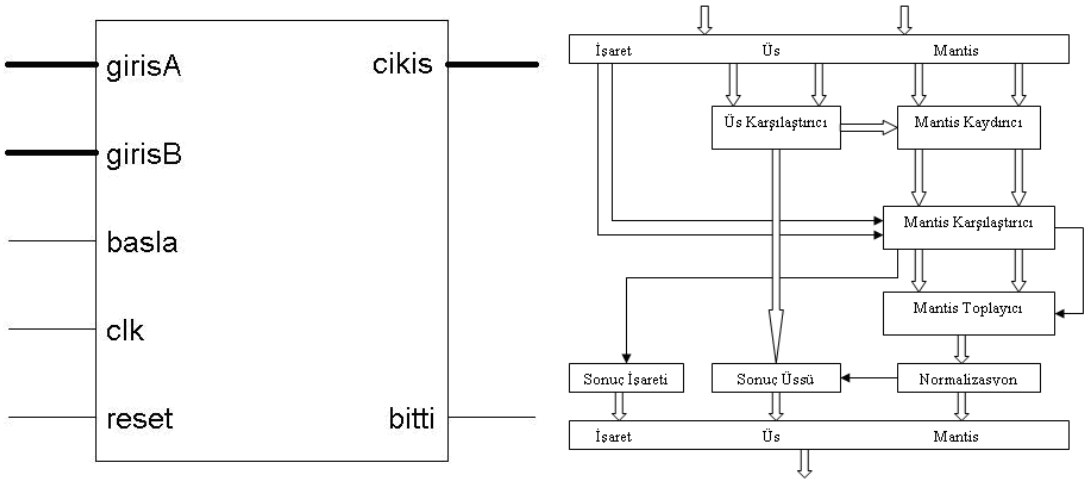
Herhangi bir sayısal filtrenin temel elemanları toplama ve çarpma bloklarıdır. Kayan nokta aritmetiğinde toplama ve çarpma yapan devreler parametrik olarak VHDL donanım tanımlama dili ile tasarlanmıştır. Devrelerin sentezlenmesi öncesinde işlem görecek kayan noktalı sayıların, toplam bit sayısı, mantis bit sayısı, üs bit sayısı kullanıcı tarafından istenildiği gibi belirlenebilir. Mantisin saklı biti işleme devre içinde dahil edilmektedir. Doğal olarak belirlenen bit sayılarına göre devreler, farklı bit sayıları için, çalışma frekansı ve kapladığı alan bakımından farklılıklar gösterecektir.

4.1 Toplama Devresi

Toplama devresi gerçeklenirken toplama algoritması olarak bölüm 3.2'de ki bilgiler referans alınmıştır. Devrenin birim yapısı ve blok diyagramı Şekil 4.1'de görülmektedir. Parametrik olan devrenin, işlem yapacağı sayıların bit uzunluğunu belirlemek için, en üst birimin VHDL kodunda bulunan "generic" kısmındaki değişkenlere değer vermek yeterlidir. Bu değişkenlerden, x (toplam bit sayısı-1)'i, n (mantis bit sayısı-1)'i, m de (üs bit sayısı-1)'i simgelemektedir.

Tasarlanan devrede başla girişinin '1' olmasıyla, ilk önce üsler karşılaştırılmakta, sonra mantisler oluşan farka göre kaydırılıp karşılaştırılmakta ve sonuç işareti belirlenmekte, daha sonra mantisler toplanmakta ya da çıkarılmakta, son olarak da mantis normalize edilip sonuç üssünde varsa gerekli değişiklikler yapılarak işlem tamamlanmaktadır. Fakat girişlerden herhangi biri sıfırsa bu işlemler yapılmadan çıkışa hemen sıfır olmayan giriş verilmektedir [5].

Devrede dört tane alt birim bulunmaktadır. Bu birimler: Üs karşılaştırıcı, mantis kaydırıcı, mantis karşılaştırıcı ve mantis toplayıcıdır. Bütün birimlerde eşzamanlı reset ve clk ile sembolize edilmiş saat girişi ortaktır.

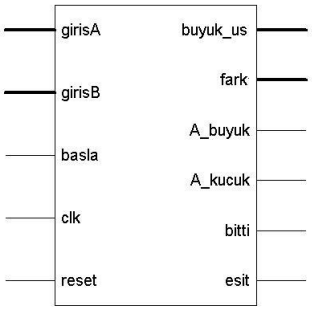


(a) (b)

Şekil 4.1: Toplama devresinin (a) Birim yapısı (b) Blok diyagramı

4.1.1 Üs Karşılaştırıcı

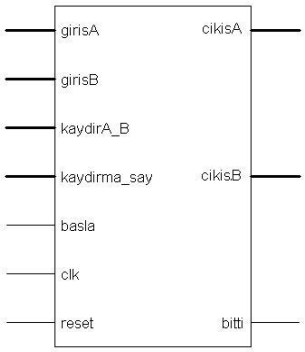
Üs karşılaştırıcı biriminin giriş ve çıkışları Şekil 4.2'de görülmektedir. Devrede reset girişinin '1' olmasıyla bütün çıkışlar sıfırlanır. Devreye başla sinyali geldiğinde A ve B girişlerindeki üs değerleri alınarak çıkarılır. Çıkarma işlemi seri olarak gerçekleştirilir. Çıkarma işlemi sonucunda hangi üssün daha büyük olduğu belirlenir. İşlem bittiğinde bitti çıkışı '1' olur ve A girişi büyükse A_buyuk çıkışı '1', aksi halde A_kucuk çıkışı '1' olur. Eğer üsler eşitse esit çıkışı '1' olur. Ayrıca çıkışa büyük olan üs ve üsler arasındaki fark da verilir.



Şekil 4.2: Üs karşılaştırıcı birimi

4.1.2 Mantis Kaydırıcı

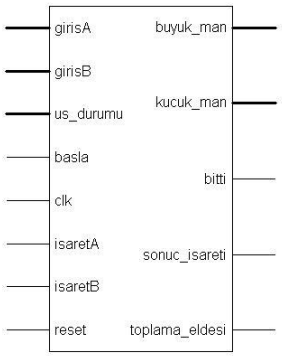
Mantis kaydırıcı biriminin giriş ve çıkışları Şekil 4.3'de görülmektedir. Devrenin başla girişine üs karşılaştırıcının bitti çıkışı bağlıdır. Üs karşılaştırma işlemi bittikten sonra mantis kaydırma işlemi direkt olarak başlar. Hangi mantisin ne kadar kaydırılacağına kaydirA_B ve kaydirma_say girişlerine göre karar verilir. Bu girişlere üs karşılaştırıcının fark, A_buyuk, A_kucuk ve esit çıkışları bağlıdır. Mantislerin saklı bitleri kaydırma işlemi başlamadan birim içerisinde mantislere dahil edilir. Kaydırma işlemi bittiğinde mantis kaydırıcının bitti çıkışı bir olur ve çıkışa mantislerin kaydırılmış halleri verilir.



Şekil 4.3: Mantis kaydırıcı birimi

4.1.3 Mantis Karşılaştırıcı

Mantis karşılaştırıcı biriminin giriş ve çıkışları Şekil 4.4'de görülmektedir. Devrenin başla girişine mantis kaydırıcının bitti çıkışı bağlıdır. Mantis kaydırma işlemi bittikten hemen sonra mantis karşılaştırma işlemi başlar. Mantisler her durumda karşılaştırılmaz. Mantislerin hangi durumlarda karşılaştırılacağına, us_durumu, isaretA ve isaretB girişlerinin durumlarına göre karar verilir.

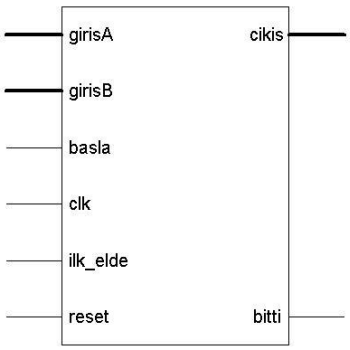


Şekil 4.4: Mantis karşılaştırıcı birimi

Karşılaştırma işlemi bittiğinde mantis karşılaştırıcının bitti çıkışı ‘1’ olur ve buyuk_man çıkışına büyük mantis, kucuk_man çıkışına küçük mantis verilir. Toplama_eldesi çıkışı, sayılar aynı işaretli ise ‘0’ aksi durumda ‘1’ olur. Ayrıca sonuç işareti de bu birim içinde belirlenir.

4.1.4 Mantis Toplayıcı

Mantis toplayıcı biriminin giriş ve çıkışları Şekil 4.5’de görülmektedir. Mantis karşılaştırıcının bitti çıkışı, mantis toplayıcının başla girişine bağlıdır. Mantis karşılaştırma işlemi bittikten hemen sonra mantis toplama işlemi yapılır. İlk elde girişine mantis karşılaştırıcının toplama_eldesi çıkışı bağlıdır. Bu giriş ‘1’ iken sayıların işaretleri ters demektir ve çıkarma işlemi gerçekleşir, tersi durumda ise toplama işlemi gerçekleşir. Toplama veya çıkarma seri olarak gerçekleşir ve tamamlandığında bitti çıkışı ‘1’ olur, toplam çıkışa verilir.



Şekil 4.5: Mantis toplayıcı birimi

4.1.5 Normalizasyon

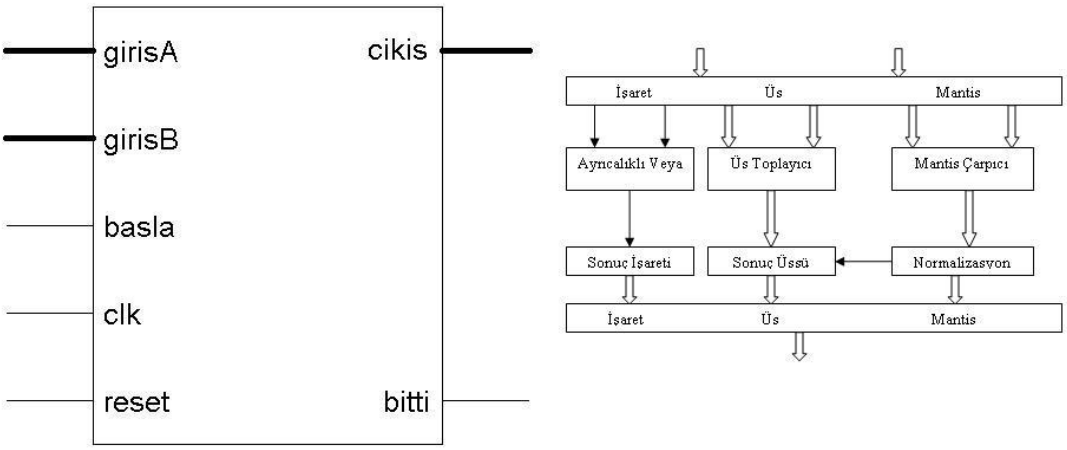
Mantisin normalize edilmesi ve sonuç üssünün düzenlenmesi işlemleri için ayrı bir alt birim tasarlanmamış olup, bu işlemler en üst birimde gerçekleştirilmektedir. Mantis toplayıcının bitti çıkışı bir olduktan sonra toplam mantisi kaydırılarak normalize edilir. Eğer mantis sağa kaydırılırsa üs karşılaştırıcının buyuk_us çıkışından alınan değer arttırılır, aksi durumda ise azaltılır. Bu yolla normalize edilmiş sonuç mantisi ve sonuç üssü elde edilir. Bu işlemden sonra da sonuç işareti, sonuç üssü ve sonuç mantisi birleştirilerek çıkışa verilir.

4.2 Çarpma Devresi

Çarpma devresi gerçeklenirken çarpma algoritması olarak bölüm 3.3'de ki bilgiler referans alınmıştır. Devrenin birim yapısı ve blok diyagramı Şekil 4.6'de görülmektedir. Parametrik olan devrenin, işlem yapacağı sayıların bit uzunluğunu belirlemek için, en üst birimin VHDL kodunda bulunan "generic" kısmındaki değişkenlere değer vermek yeterlidir. Bu değişkenlerden, x (toplam bit sayısı-1)'i, n (mantis bit sayısı-1)'i, m de (üs bit sayısı-1)'i simgelemektedir.

Tasarlanan devrede başla girişinin '1' olmasıyla üsler toplanır ve üs toplamından yanlılık çıkarılır. Aynı anda da mantisler çarpılmaya başlar. İki işlem de bittiği zaman mantisler normalize edilir ve sonuç üssü eğer gerek varsa düzenlenir. Sonuç işareti sayıların işaret bitlerinin ayrıcalıklı veya (exor) işlemine tabi tutulmasıyla bulunur. Böylece işlemler tamamlanmış olur. Fakat girişlerden herhangi biri sıfırsa bu işlemler yapılmadan çıkışa hemen sıfır verilir [6].

Devrede iki tane alt birim bulunmaktadır. Bu birimler üs toplayıcı ve mantis çarpıcıdır. Bütün birimlerde eşzamanlı reset ve clk ile sembolize edilmiş saat girişi ortaktır.

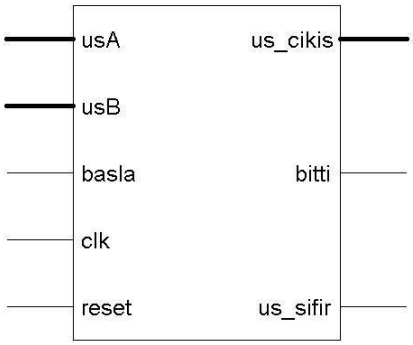


(a) (b)

Şekil 4.6: Çarpma devresinin (a) Birim yapısı (b) Blok diyagramı

4.2.1 Üs Toplayıcı

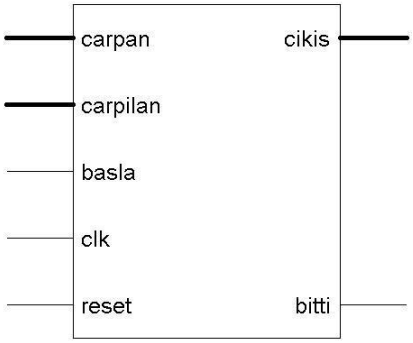
Üs toplayıcının birim yapısı Şekil 4.7'de görülmektedir. Devrenin başla girişi '1' olduğu zaman devre girişindeki üsler içeri alınır ve toplanır. Toplama işleminden sonra devre içinde üs bit sayısına göre belirlenen yanlılık toplamdan çıkarılır. Devrede toplama ve çıkarma işlemleri seri olarak gerçekleştirilmektedir. Eğer çıkarma sonucunda üs sıfırdan küçük çıkarsa us_sifir çıkışı '1' olur ve us_cıkıs '0' yapılır. Çıkarma işleminin bitmesiyle devrenin bitti çıkışı da '1' olur ve sonuçlar çıkışa verilir.



Şekil 4.7: Üs toplayıcı birimi

4.2.2 Mantis Çarpıcı

Mantis çarpıcının birim yapısı Şekil 4.8'da görülmektedir. Devrenin başla girişi '1' olduğunda mantisler devre içine alınarak çarpılır. Mantislerin saklı biti mantislere bu birim içinde eklenir. Çarpma işlemi bit seri çarpma algoritması kullanılarak gerçekleştirilmiştir [4]. Çarpma işlemi bittiğinde devrenin bitti çıkışı '1' olur ve çarpım çıkışa verilir.



Şekil 4.8: Mantis çarpıcı birimi

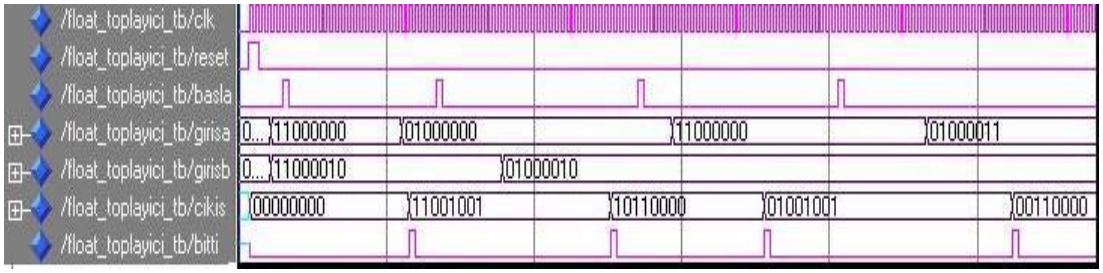
4.2.3 Normalizasyon

Mantisin normalize edilmesi ve sonuç üssünün düzenlenmesi işlemleri için ayrı bir alt birim tasarlanmamış olup, bu işlemler, toplama devresinde olduğu gibi, en üst birimde gerçekleştirilmektedir. Çarpma ve toplama işlemleri bittiğinde, devrede yine kaydırma işlemi ile mantis normalize edilmekte ve gerekirse üs de düzenlenerek işlemler tamamlanmaktadır.

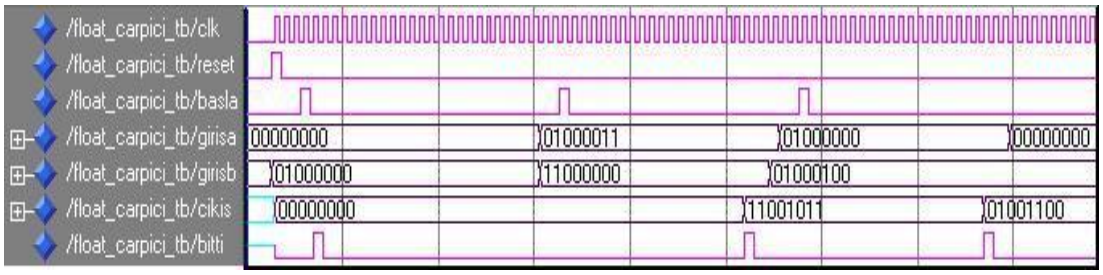
4.3 Benzetim Sonuçları

Toplama ve çarpma devreleri Xilinx ISE 7.1i programında tasarlamış, sentezlenmiş ve yine bu programla devrelerin yerleştirme ve yollandırma işlemleri gerçekleştirilmiştir. Devrelerin davranışsal ve yerleştirme ve yollandırma sonrası benzetimleri ise ModelSim EXE III 6.0a programında yapılmıştır. Şekil 4.9(a)'da toplama devresinin, Şekil 4.9(b)'de ise çarpma devresinin benzetim sonuçları görülmektedir. Sonuçların daha kolay incelenebilmesi için kayan nokta gösterimindeki sayılar 8 bitlik (1 işaret biti, 4 üs biti, 3 mantis biti) alınmıştır.

Davranışsal ve yerleştirme ve yollandırma sonrası benzetimlerin sonuçları aynı çıkmış ve devrelerin doğru bir şekilde çalıştığı gözlenmiştir.



(a)



(b)

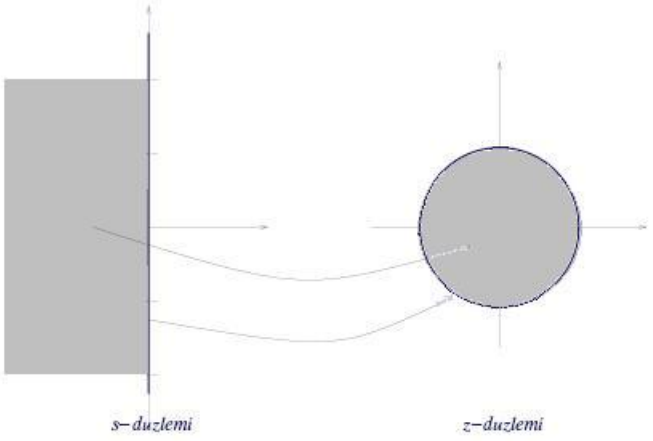
Şekil 4.9: Benzetim sonuçları (a) Toplama devresi (b) Çarpma devresi

5. SAYISAL SONSUZ İMPULS YANITLI (IIR) FİLTRE

5.1 Sayısal IIR Filtre Teorisi

Sayısal IIR filtrelerin transfer fonksiyonlarının elde edilmesinde kullanılan yöntem analog filtre transfer fonksiyonlarının dönüşümler yoluyla sayısala uyarlanmasıdır. İlk önce istenilen karakteristiği sağlayan uygun bir analog filtre transfer fonksiyonu H(s) bulunur. Daha sonra da uygun bir dönüşüm yardımıyla H(s) transfer fonksiyonundan H(z) transfer fonksiyonuna geçilir. Bu dönüşümlerin sağlaması gereken iki tane koşul vardır (Bkz Şekil 5.1):

(i) S-düzlemindeki sanal eksen z-düzlemindeki birim daireye dönüşmelidir.
(ii) Sol yarı s-düzlemi z-düzleminde birim dairenin içine düşmelidir.



Şekil 5.1: S-düzleminden z-düzlemine geçiş

Analog IIR filtrelerin transfer fonksiyonlarını dönüştürmek için kullanılan üç ana yöntem vardır:

1. Sayısal integrasyon metodu
2. Değişmez impuls yanıtı metodu
3. Bilineer dönüşüm metodu

Sayısal IIR filtrelerin transfer fonksiyonu en genel halde (5.1) eşitliğinde olduğu gibidir.

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{i=0}^{L} b_i z^{-i}}{\sum_{m=0}^{M} a_m z^{-m}} \quad (5.1)$$

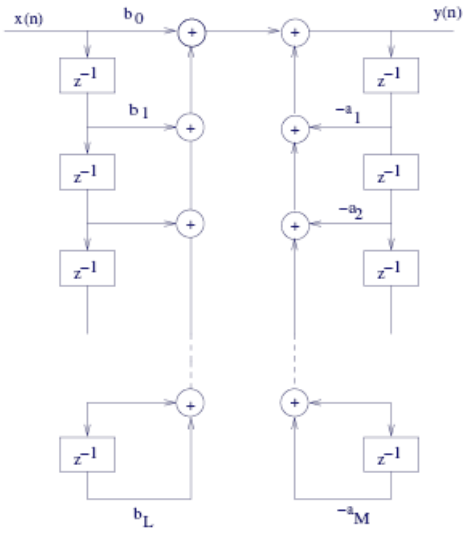
Bu fonksiyonda katsayılar, a_0 katsayısı 1 olacak şekilde düzenlenir ve $x(z)z^{-1} = x(n-1)$ dönüşümü uygulanırsa, (5.2) eşitliğindeki fark denklemi elde edilir [7-9].

$$y(n) = b_0 x(n) + b_1 x(n-1) + \ldots + b_L x(n-L) - a_1 y(n-1) - a_2 y(n-2) - \ldots - a_M y(n-M) \quad (5.2)$$

5.2 Sayısal IIR Filtrenin Gerçeklenmesi

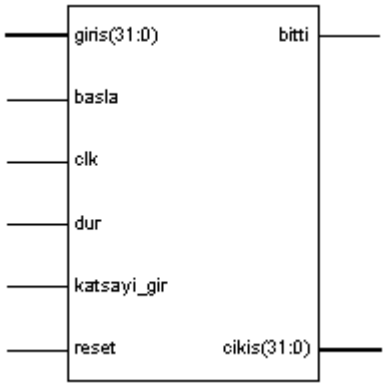
Filtre devresi gerçeklenirken devrenin (5.2) eşitliğindeki fark denklemini sağlayacak şekilde olması amaçlanmıştır. Toplama ve çarpma devreleri gibi filtre de parametrik olarak tasarlanmıştır. Filtrenin VHDL kodu Ek-A'da verilen CD'de mevcuttur. Filtrenin kaçıncı dereceden olacağını ve kaç bit uzunluğunda sayılarla işlem yapacağını belirlemek için en üst birimin VHDL kodunda bulunan "generic" kısmındaki değişkenlere değer verilmesi yeterlidir. Bu değişkenlerden, d filtre derecesini, x (toplam bit sayısı-1)'i, n (mantis bit sayısı-1)'i, m de (üs bit sayısı-1)'i simgelemektedir.

Devre gerçeklenirken fark denklemini sağlamak için Şekil 5.2'de görülen direkt-I yapısı kullanılmıştır [9].



Şekil 5.2: Direkt-I filtre yapısı

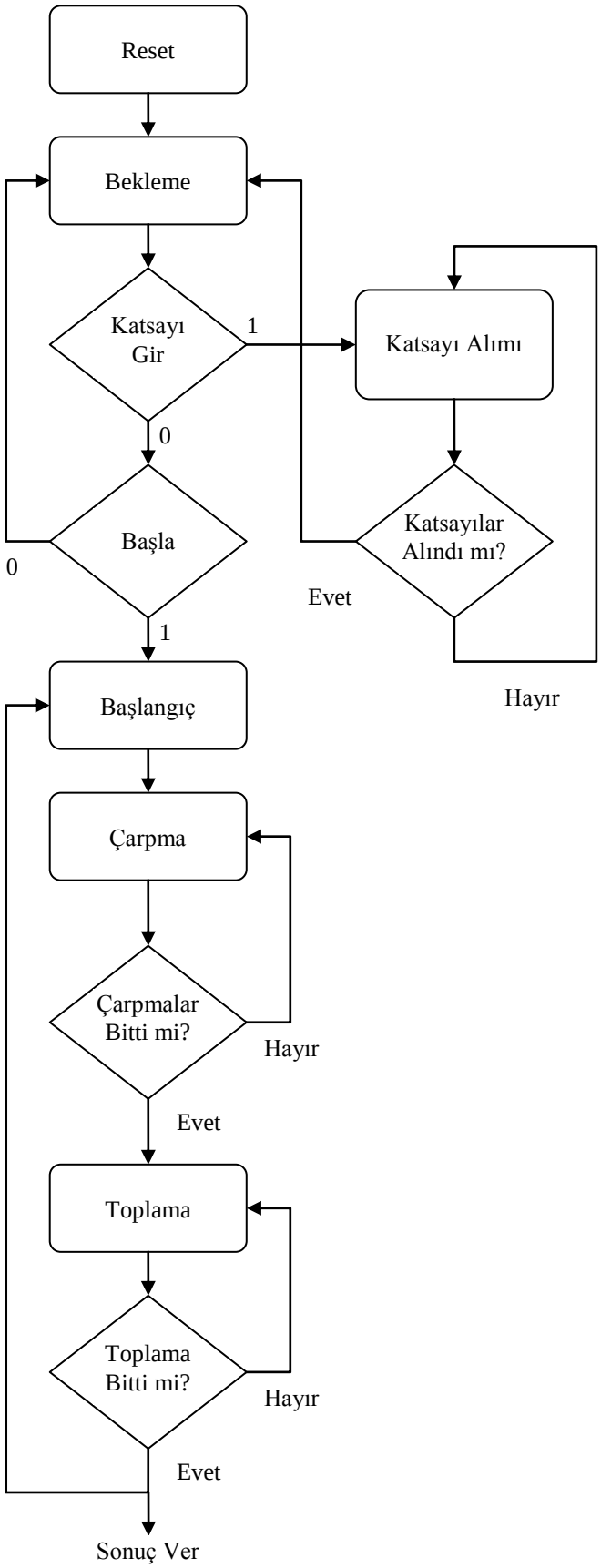
Filtrenin birim yapısı Şekil 5.3'de görülmektedir. Devrenin iki tane alt birimi vardır. Bunlar toplama ve çarpma devreleridir. Denklem (5.2)'deki çıkarma işlemleri ise çıkışların geçmiş değerlerini çarpan katsayıların ters işaretlilerinin kullanılmasıyla gerçekleştirilmiştir.



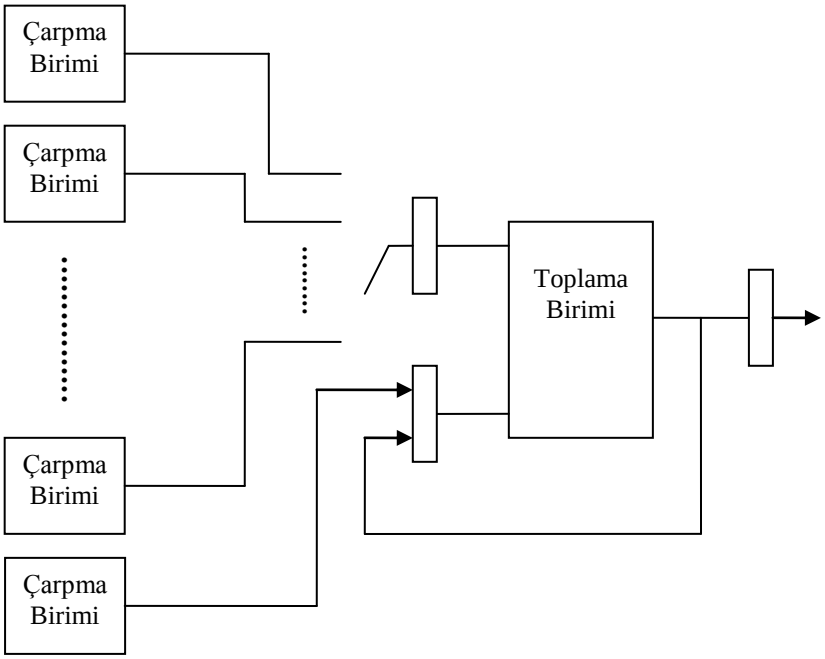
Şekil 5.3: Filtre birimi

Şekil 5.4'de filtreleme işleminin akış diyagramı görülmektedir. Diyagramın karışık olmaması için dur girişinin etkisi diyagrama dahil edilmemiştir. Devre ilk çalışmaya başlarken reset girişi '1' olmalıdır. Böylece bütün çıkışlar ve devre içindeki saklayıcılar sıfırlanır, filtre de bekleme durumuna geçer. Filtre bu durumda iken katsayi_gir girişi '1' olursa filtre katsayı alımı durumuna geçer. Bu durumda iken gelen girişler katsayı olarak değerlendirilir ve katsayılar için ayrılmış saklayıcılara yazılır. Bütün katsayılar alındıktan sonra devre kendi kendine tekrar bekleme durumuna geçer. Bekleme durumunda iken başla girişi '1' olursa devre başlangıç durumuna geçer ve bu durumda iken girişteki veriyi alarak doğrudan çarpma durumuna geçer. Devrede bütün çarpma işlemleri paralel olarak gerçekleştirilmektedir (Şekil 5.5). Bütün çarpma birimleri işlemlerini bitirdikleri zaman ise devre toplama durumuna geçer. Devrede tek toplayıcı birimi bulunmaktadır (Şekil 5.5). Bu birim ilk önce ilk iki çarpımı toplar, daha sonra da bu toplamla bir sonraki çarpımı toplar ve bu şekilde devam eder. Bu işlem bütün çarpımlar toplanana kadar sürer. Sonuç toplamı oluştuğunda ise devre bitti çıkışı '1' olur, sonuç

dışarıya verilir ve devre tekrar başlangıç konumuna geçer. Bekleme durumu hariç diğer bütün durumlarda dur girişi ‘1’ olursa devre bekleme durumuna geçer. Eğer devreye dur girişi hiç gelmezse, devre sürekli olarak çalışmaya devam eder.



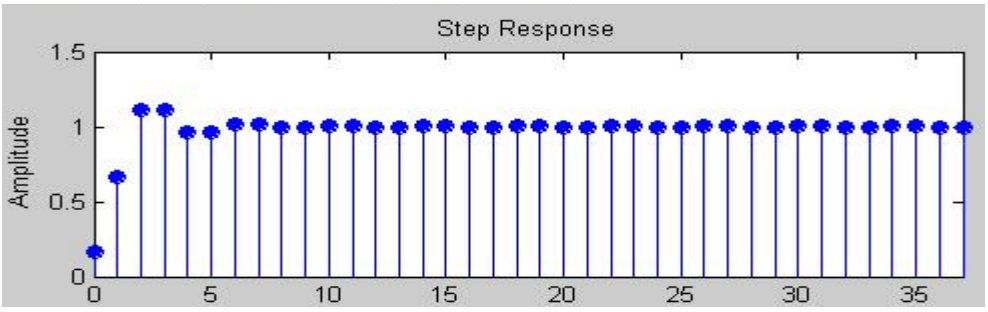
Şekil 5.4: Filtre akış diyagramı



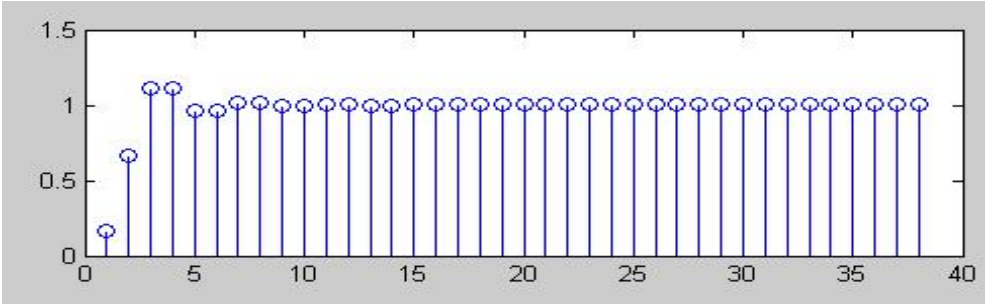
Şekil 5.5: Filtre blok diyagramı

5.3 Benzetim Sonuçları

Filtre devresi Xilinx ISE 7.1i programında tasarlamış, sentezlenmiş ve yine bu programla devrenin yerleştirme ve yollandırma işlemleri gerçekleştirilmiştir. Devrenin, davranışsal ve yerleştirme ve yollandırma sonrası benzetimleri ise ModelSim EXE III 6.0a programında yapılmıştır. Benzetimlerde üçüncü dereceden alçak geçiren Butterworth tipi bir filtre kullanılmıştır. Filtrenin işlem yaptığı sayılar ise IEEE standardında tek duyarlı kayan noktalı sayılardır. Filtre katsayıları MATLAB programı vasıtasıyla hesaplanmıştır. Filtrenin çalışmasını denetlemek için devrenin birim basamak ve birim darbe cevaplarına bakılmıştır. Çıkan sonuçların doğruluğu ise yine MATLAB programında çizdirilen ideal filtreye ait birim basamak ve birim darbe cevaplarının, devre çıktılarıyla karşılaştırılmasıyla sınanmıştır. Filtreye ait davranışsal ve yerleştirme ve yollandırma sonrası benzetimlerde aynı sonuçlar elde edilmiştir. Benzetim sonuçları Ek-A'da verilen CD'de mevcuttur. Şekil 5.6'da birim basamak cevabı sonuçları, Şekil 5.7'de de devrenin birim darbe cevabı sonuçları görülmektedir.

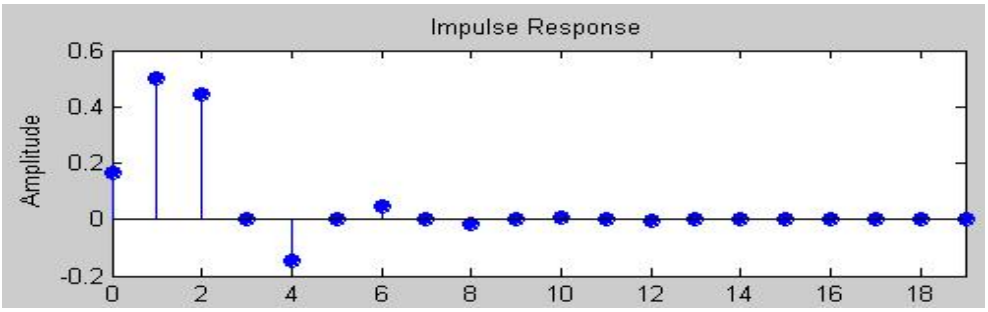


(a)

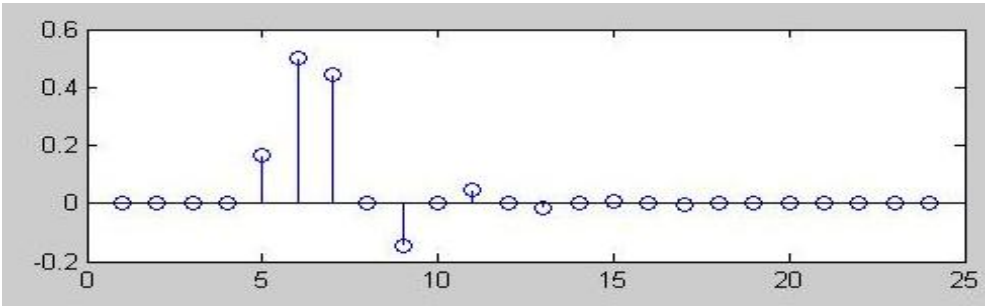


(b)

Şekil 5.6: Birim basamak cevabı (a) MATLAB çıktısı (b) Devre çıktısı



(a)



(b)

Şekil 5.7: Birim basamak cevabı (a) MATLAB çıktısı (b) Devre çıktısı

6. SONUÇLAR

FPGA üzerinde tasarlanan üçüncü dereceden filtre devresinin, yerleştirme ve yollandırma sonrası saat frekansı en fazla 76 MHz olmuştur. Fakat bir sorunla karşılaşmamak için yerleştirme ve yollandırma sonrası benzetimde yaklaşık 40 MHz'lik bir saat frekansı kullanılmış ve devre sorunsuz bir şekilde çalışarak doğru sonuçlar vermiştir. Tablo 6.1'de üçüncü dereceden tek duyarlı kayan nokta aritmetiğinde devrenin kapladığı alana ilişkin bilgiler görülmektedir.

Tablo 6.1: FPGA'da kullanılan alan

Lojik Birimler	Mevcut Sayı	Kullanılan Sayı	Yüzde
Dilimler	12288	4111	%33
Dilimlerdeki Flip-Floplar	24576	4358	%17
Dört Girişli LUT'lar	24576	6270	%25
IOB'ler	158	69	%43
Saat	4	1	%25

Yapılan denemelerde FPGA alanı en üst düzeyde kullanıldığında filtre derecesi en fazla on dört olmuştur. Fakat bu durumda çarpım sayısı artacağından ve çarpımları toplamak için tek toplayıcı birimi kullanıldığından, bir çıkışın oluşması oldukça uzayacaktır. Yine de çıkışın oluşma süresinin önemsiz olduğu sistemlerde bu durum bir sorun yaratmayacaktır.

FPGA içinde kullanılan alandan daha fazla yararlanmak için, filtre yapısı direkt-I yerine direkt-II seçilebilir. Böylece geçmiş değerleri tutmak için gerekli saklayıcı sayısı yarıya inmiş olur. Filtre yapısı için evrik direkt-I formu da kullanılabilir. Bu yolla hem saklayıcı sayısı hem de gerekli toplama işlemi sayısı azaltılmış olur [8].

Filtre de çıkış oluşma sürecinin hızlanması için ise toplama ve çarpma işlemleri için seri algoritmalar yerine daha hızlı sonuç veren algoritmalar kullanılabilir.

Tasarlanan devrenin bir özelliği de FPGA programlandıktan sonra filtre tipinin katsayi_gir girişi '1' yapılıp yeni katsayılar yüklenerek değiştirilebilir olmasıdır. Böylece değişik tipte ve özellikteki filtrelerin tek bir devre üzerinde gerçeklenebilmesi ve kullanım anında filtre tipinin değiştirilmesi mümkün olmaktadır.