APKD' LERİNDE FIR SÜZGEÇ TASARLAMAK İÇİN GELİŞTİRİLEN VHDL KODU VE ŞEMATİĞİ

İbrahim Emrah TABARU[1] , Sabri BİCAKCI[2] , Nurhan KARABOĞA[3]

[1]Elektrik ve Elektronik Mühendisliği Bölümü, Erciyes Üniversitesi, Kayseri

[1]e-posta : 1030225157@erciyes.edu.tr [2]e-posta: 1030225197@erciyes.edu.tr,
[3]e-posta: nurhan_k@erciyes.edu.tr

Anahtar Sözcükler: FIR Süzgeç, APKD, VHDL Kodu

ÖZET

Elektronik cihazların birçoğunda kullanılan temel elemanlardan birisi sayısal süzgeçlerdir. Sayısal süzgeçlerin donanımsal olarak gerçekleştirilmesinde farklı yaklaşımlar mevcuttur. APKD (Alan Programlanabilir Kapı Dizisi – FPGA, Field Programmable Gate Array Logic)' ler, mikro denetleyiciler veya opamplar vb. kullanılarak sayısal süzgeçler gerçekleştirilebilmektedir. Bu çalışmada APKD ile sayısal FIR süzgeç tasarımı konusu incelenerek düşük güç tüketimine ve iyi bir performansa sahip olan APKD' lerinin sayısal süzgeçlerin donanım olarak gerçekleştirilmesinde daha az kapı elemanı kullanılmasına imkan sağlayan VHDL kodu ve şematiği geliştirilmiştir. Böylece aynı zamanda analog sayısal çeviricinin çözünürlüğüne bağlı olarak kullanılacak bit sayılarında da bir esneklik sağlanmıştır.

1. GİRİŞ

Ayrık zamanlı bir süzgecin dürtü yanıtı sadece sonlu sayıda sıfırdan farklı değere sahip ise süzgeç sonlu dürtü yanıtlı (Finite Impulse Response – FIR) süzgeç olarak adlandırılmaktadır. Dürtü yanıtı sonlu olmayan ve sonsuza kadar devam eden süzgeçler, sonsuz dürtü yanıtlı (Infinite Impulse Response –IIR) süzgeç olarak adlandırılmaktadır. Bir FIR süzgeç çıkışı şimdiki ve geçmişteki giriş değerlerinin ağırlıklı toplamından oluşacak bir şekilde tanımlanabilir. Bir çok elektronik cihazda sayısal süzgeçler kullanılmaktadır. Sayısal süzgeçler, simülasyonlarla gerçekleştirilebilirler fakat asıl önemli olan süzgeçlerin donanımsal olarak da gerçekleştirilebilmesidir. Literatürde sayısal süzgeç tasarımında APKD' lerinin kullanılmasıyla ilgili birçok çalışma vardır[1-8].

APKD ile gerçekleştirilen tasarımlarda güç tüketimlerinin düşük ve performanslarının yüksek olduğu bilinmektedir. APKD ile yapılan tasarımlar oldukça esnek olabilmektedir ve APKD'lerin bir işlemi gerçekleştirme hızları oldukça yüksektir. Ayrıca APKD' leri ile aritmetik birimler kolayca gerçeklenebilmektedir.

Sayısal süzgeçler ayrık zamanlı işaretlerin süzgeçlenmesi için kullanılan sayısal sistemlerdir. Sayısal süzgeçler sayısal sistemlerin genel üstünlüklerini taşımakta ve özellikle süzgeç karakteristiğinin basit bir şekilde değiştirilebilmesi veya uygulanması mümkün olduğundan yaygın olarak kullanılmaktadır. Bu çalışmada, FIR süzgeçlerin APKD kullanılarak tasarımı konusu incelenmiştir. Sayısal FIR süzgeçler her zaman kararlıdır ve genel olarak aşağıdaki transfer fonksiyonu ile karakterize edilirler:

$$H(z) = \sum_{n=0}^{N} a(n) z^{-n} \qquad (1)$$

1. eşitlikte N süzgeç derecesini, a(n) ise süzgeç katsayılarını tanımlamaktadır.
Çalışmanın ikinci bölümünde APKD' leri hakkında bilgi verilmiştir. Üçüncü bölümde APKD' lerinin sayısal süzgeç tasarımında nasıl kullanıldığı konusu anlatılarak bu çalışmada geliştirilen VHDL kodu ve şematiği ayrı ayrı anlatılarak sonuçlar yorumlanmıştır.

2. ALAN PROGRAMLANABİLİR KAPI DİZİSİ (APKD)

Programlanabilir mantıksal elemanlar, bağımsız olarak oluşturulabilen mantıksal kapılar, kaydediciler ve bunları birbirine bağlayan programlanabilir bağlantılardan oluşmaktadırlar.

Bu elemanların ilki, 1975 yılında Monolithic Memories Inc. (MMI) tarafından geliştirilmiş olan PAL (Programmable Array Logic – Programlanabilir Dizi Lojik) elemanıdır[9]. PAL' ler de, Boolean fonksiyonları, De Morgan kuralları kullanılarak çarpımların toplamı yada toplamların çarpımı şeklinde gerçekleştirilebilirler. PAL yapısında girişlerin kendisi ve terslenmiş halleri AND kapılarına girmekte ve bu kapıların çıkışları da OR kapılarına uygulanmaktadır. Bu yapılar kullanılarak fonksiyon gerçekleştirilmektedir. PAL yapılarında OR kapıları sabit iken AND kapıları programlanabilir yapıya sahiptir.

Diğer bir eleman ise PLA (Programmable Logic Array – Programlanabilir Mantıksal Dizi)'dir. Bu elemanın PAL' lerden farkı ise çıkıştaki OR kapılarının da programlanabilir olması ve böylece daha büyük fonksiyonların gerçekleştirebilmesidir. Bu yapılarda programlama işlemi bağlantı hatlarının koparılması yada bağlı kalması sağlanarak gerçekleştirilmektedir. Daha karmaşık fonksiyonların gerçekleştirilebilmesi için PAL ve PLA' ların birleşiminden oluşan CPLD (Complex Programmable Logic Device – Karmaşık Programlanabilir Mantıksal Eleman)'ler geliştirilmiştir. PAL' ler bir programlayıcı ile programlanabilirken, CPLD' ler üreticinin geliştirdiği bir metotla veya bilgisayara bağlı JTAG kablo ile bir sistem yardımıyla programlanabilmektedirler. CPLD' ler de yüksek performanslı elemanlardır.

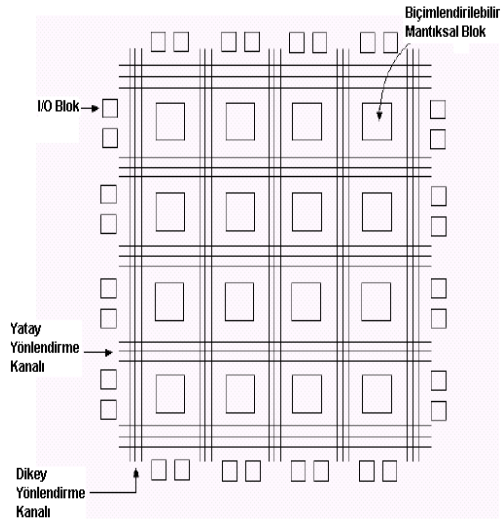
Geliştirilen alternatif programlanabilir mantıksal elemanlardan biride APKD' leridir. Bu yapıyı Xilinx firması 1984 yılında geliştirmiştir[10]. Bu eleman daha esnek bir yapıya sahiptir. APKD' nin temel parçası LUT (Look Up Table – Hafıza Tablosu)'dır. LUT bir fonksiyon jeneratörü gibi davranmakta yada bir ROM (Read Only Memory – Yalnızca Okunabilir Hafıza) veya RAM (Random Access Memory – Rasgele Erişilebilir Hafıza) olarak da ayarlanabilmektedir. Birbiri ile hızlı mantıksal bağlantısı olan hücreler içerdikleri için APKD' leri aritmetik fonksiyonları ve daha ileri DSP (Digital Signal Processing – Sayısal İşaret İşleme) uygulamaları için oldukça kullanışlıdırlar.

APKD' lerinin çoğunluğu SRAM (Static Random Access Memory – Statik Rasgele Erişilebilir Hafıza) tabanlı olduğu için standart bir SRAM kadar kolay programlanabilirler.

APKD' lerinin programlanması için kullanılan programlama dilleri HDL(Hardware Description Language – Donanım Tanımlama Dili) türü Verilog ve VHDL (Very High Speed Integrated Circuit Hardware Description Language - Yüksek Hızlı Tümleşik Devre Donanım Tanımlama Dili)'dir. Her ikisi de sayısal devrenin yazı tabanlı tanımlanmasıdır. Ayrıca C gibi üst seviyeli dillerle de biçimlendirme yapılabilir. APKD'ler birçok yöntem ile biçimlendirilebilirler. Her üretici firmanın bir yazılımı vardır ve bu yazılım ile hangi yöntem kullanılarak tasarım yapılmış ise o tasarımı APKD' ni programlayacak, yani APKD' ye yüklenebilecek hale dönüştürür. Şematik tasarım en kolay biçimlendirme metodudur. Yazılımda hazır bulunan elemanlar ve makrolar kullanılarak devre öncelikle şematik olarak oluşturulmaktadır.

Şekil 1' de bir APKD' nin iç yapısı verilmektedir[2]. Her bir biçimlendirilebilir mantık bloğu, kendi aralarında ve her bir I/O (Input/Output – Giriş/Çıkış) blok arasında yönlendirme kanalları aracılığı ile irtibatlandırılabilmektedir.

LUT yapıları mantıksal bloklar içinde bulunmaktadır. LUT, temel olarak hafıza birimleri ve bu hafıza birimlerinden hangisinin çıkışa aktarılacağını belirleyen bir çoğullayıcı yapıdan meydana gelmektedir.



Şekil 1 APKD' nin Temsili İç Yapısı

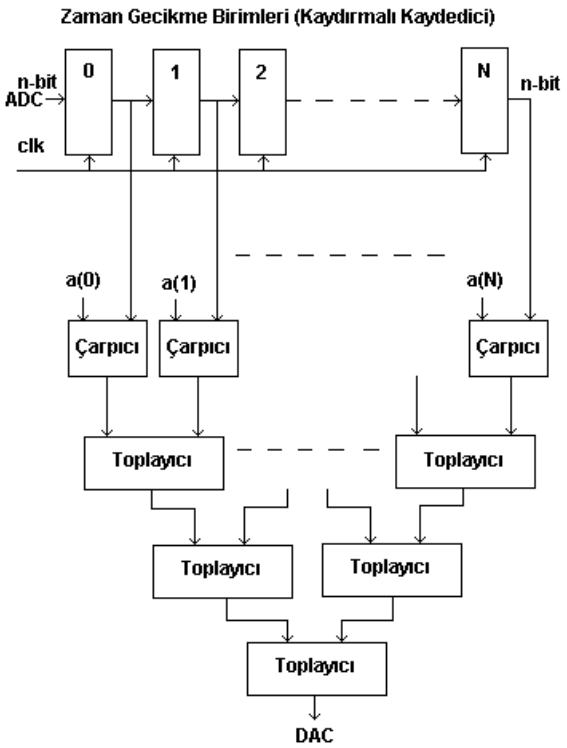
APKD' lerinin yapılarında birçok mantıksal eleman, temel hafıza birimleri ve kontrol edilebilir bağlantılı iletim yolları mevcut olduğu için sinyal işlemenin en temel fonksiyonları olan toplama, çıkarma ve gecikme fonksiyonları da kolayca gerçekleştirilebilmektedir.

3. SAYISAL SÜZGEÇ TASARIMINDA VHDL KODUNUN KULLANILMASI

APKD ile sayısal süzgeç tasarımında, sayısal süzgecin giriş, çıkış ve katsayılarını belirleyen bitlerin sayısı arttıkça, kullanılması gereken elemanın boyutları da üstel olarak artmaktadır. Bu sorunu aşmak için, bit seri tasarım mantığı kullanılmaktadır. Böylece istenen süzgeç tek bir APKD üzerinde gerçekleştirilebilmektedir. Bu yöntemin paralel tasarıma göre olumsuz yönü; n-bit paralel tasarımın bir çevrimde yaptığı işi seri tasarım n çevrimde gerçekleştirebilmektedir.

Tasarımda sistemin hızı ön planda tutulduğu için paralel mantık kullanılmaktadır. FIR süzgecin oluşturulmasında kullanılan birimlerin genel bağlantısı Şekil' 2 de görülmektedir.

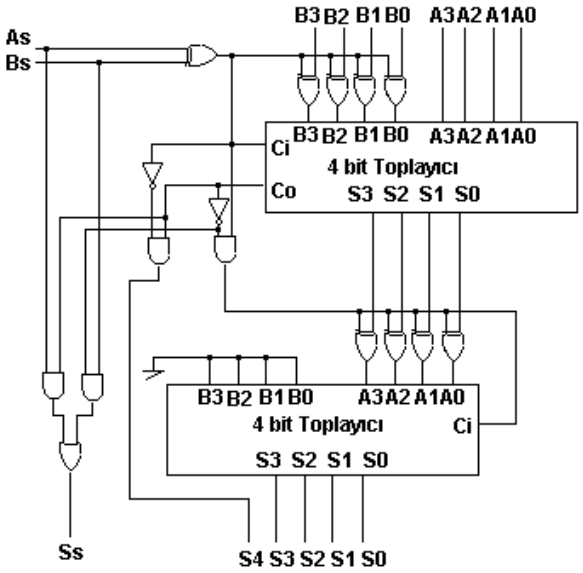
Zaman gecikme birimlerinde her bir blok n adet D tipi flip-floptan meydana gelmektedir. a(n) katsayıları da n-bittir ve sabittir. n-bitin en yüksek değerlikli olanı işaret bitidir. Bu nedenle çarpıcı ve toplayıcı birimleri bu dikkate alınarak tasarlanmıştır. Çarpıcının çıkışı *2n-1* bittir ve her bir toplayıcı katından sonra sistem çıkışı bir bit artmaktadır. Çarpıcı tasarlanırken şematik



Şekil 2 Sistemin Genel Yapısı

tasarım içinde hazır olarak bulunan dört bitlik çarpıcı modüllerden faydalanılarak n bitlik çarpıcılar elde edilmiştir. Süzgeç katsayılarının kesirli olmasının oluşturacağı problem de katsayıların, ADC (Analog Digital Convertor – Analog Sayısal Çevirici)' nin adım büyüklüğü ile sınırlı ondalıkta seçilmesi ile aşılmıştır. Eğer hassasiyet ön planda tutulmakta ise o büyüklükte çözünürlüğe sahip bir DAC (Digital Analog Convertor – Sayısal Analog Çevirici) kullanılmalıdır. Toplatıcı birimler, işaret bitlerine göre hem toplama hem de çıkartma işlemi gerçekleştirmektedir ve sonucun da gerçek değerini üretmektedirler. Her kademede bulunan toplayıcının bit sayısı bir üst kademedekinden bir bit fazladır. Şekil 3'de bu çalışmada kullanılan toplayıcı biriminin yapısı gösterilmektedir.

Bu çalışmada ADC' nin çözünürlüğü n = 8 bit olarak alınmış ve her birimin tasarımı da ona göre yapılmıştır. n = 8 olarak alındığı için çarpıcı birimin çıkışı (*2n-1=15*) 15 bit olmaktadır. Bundan dolayı çarpıcıdan hemen sonra gelen toplayıcı birim 15 bit olarak düzenlenmiştir ve bu birimi takip eden toplayıcılar 16, 17 ve 18 bitliktir. Çünkü toplayıcı birimlerin çıkışları giriş sayısının bir bit fazlası olmaktadır. DAC' ın çözünürlüğü 8-bit seçildiği için de 4 adet toplayıcı birimi gerekmektedir. Sistem girişi n-bit iken sistem çıkışı 2n' den fazla olmaktadır. Bu sistemde ondalıklı ve tam kısmın daha az önemdeki bitleri atılarak bit sayısı n-bit'e indirgenebilmektedir.



Şekil 3 Toplayıcı Birim

```
library ieee;
use ieee.std_logic_1164.all;_
use ieee.math;

entity suzgec is

        port    ( x : in std_logic_vector ( 7 downto
0);
                clk : in bit;
          z : out std_logic_vector(7downto0);
          i: integer );
end entity;
arhitecture behavior of suzgec is
begin
process(clk)
an='0001001';
signal s: integer range 0 to 255;
signal y: integr  range 0 to 255;
if clk='1' and clok'event then
for i in 1 to 255 loop

s(i)=x;
y(i)=an*s(i);
z=0+y(i);
an=z/x;
end loop
end process
end suzgec;
```

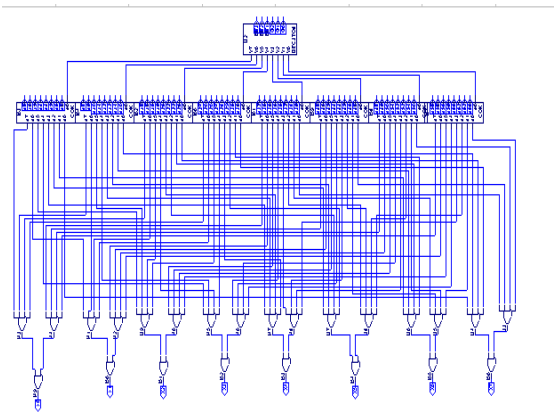
Şekil 4 Tasarlanan VHDL Kodu

Bu tasarımın VHDL kodu en temel hali ile Şekil 4'de verilmiştir. Bilgisayarda hesaplanan a(n) katsayısı,

girişlerin uygulandığı aynı analog sayısal çevirici de değeri alınmış ve çıkan sayısal değer VHDL kodu içinde kullanılmıştır.

Şematik tasarımın aksine, kod ile yapılan tasarımda bit sayılarını belirlemenin daha kolay ve esnek olduğu görülmektedir. n, 8 yerine 256 olarak alınmıştır. Çalışmada n analog sayısal çeviricinin 8 bit olmasına bağlı giriş ve a(n)'lerimiz 8 bit alınmıştır. Çarpma ve toplama öncesi işaretlerin tanımlanması ise VHDL dilinin yazım özelliğidir. VHDL kodunda ilk a(n) değerini biz verdikten sonra program a(n)'leri her 256'lık döngüden sonra değiştirmektedir. Bu da bize sonuca daha kısa sürede varmamız için bir avantaj sağlamaktadır. Yazılımla yapılan tasarımda, şematik tasarıma göre dörtte bir daha az kapı elemanı kullanmaktadır.

Şekil 5' de APKD'lerinde FIR süzgeç tasarlamak için geliştirilen VHDL şematik gösterimi çarpma birimi öncesi bir ara yapı ile 64 bitten sekiz bite bir zaman birimlerini seçme elemanı ile geçilir. Bu ara birim bize paralel ve seri arası bir tasarım sunar. Paralel mantıktaki hız korunurken bit azaltılarak seri tasarımdaki gibi eleman sayısında bir azalma yakalanır.



Şekil 5 Tasarlanan Seçme Biriminin VHDL Şematik Gösterimi

Tablo 1'de bu çalışmada kullanılan seçme birimi elemanları verilmektedir.

Tablo 1 Seçme Birimi Elemanları

Adet	Eleman
1	3x8 Kod Çözücü
8	8 bit 1 zaman gecikme birimi
16	4 – 1 OR (veya) kapısı
8	AND (ve) kapısı

4. SONUÇ

Bu çalışmada FIR tipi sayısal süzgecin APKD ile donanımsal olarak gerçekleştirilmesinde kullanılabilecek bir VHDL kodu ve şematiği geliştirilerek şematik ve VHDL kod tasarımı için esneklik sağlayabilecek alternatif bir çözüm sunulmuştur. Kod ile yapılan tasarımda bit sayılarını belirlemek daha kolay olmaktadır ve daha az kapı elemanı kullanılmaktadır. VHDL kodu ile çalışmanın bir diğer avantajıda sonuca kısa sürede ulaşılabilmektedir. Elde edilen kodlar Xilinx XC4000 ve bu APKD' nin alan kapasitesine sahip diğer APKD' leri programlamada kullanılabilmektedir. Tasarımda esnek bir yapı kullanıldığı için bit sayısı kolayca arttırılabilmektedir.

5. KAYNAKLAR

[1] Louzao J., Paz S., Tejera D., Bellora G., Langwagen G., Architectural desing of a programmable cell for the implementation of a filter bank on FPGA. Microelectronics Reliability, Vol 44, Iss 4, pp. 683-695, 2003.

[2] Re M., Cardarilli G. C., Re A. D., Lojacona R., FPGA Implementation of a Demux Based on a Multirate Filter Bank. ISCAS 2000 – IEEE International Symposium on Circuits and Systems, pp. 353-356, 2000.

[3] Lee H., Sobelman G. E., Performance evaluation and optimal design for FPGA-based digit-serial DSP functions. Computers and Electrical Engineering, Vol 29, Iss 2, pp. 357-377, 2003.

[4] Bates G. L., Nooshabadi S., FPGA Implementation of a Median Filter. IEEE TENCON –IEEE Region 10 Annual Conference, Speech and Image Technologies for Computing Telecommunications, Vol 2, pp. 437-440, 1997.

[5] Dick C., Haris F., High-Performance FPGA Filter Using Sigma-Delta Modulation Encoding, IEEE International Conference On Acoustics, Speech, and Signal Processing, , Vol 4, pp 2123-2126, 1999.

[6] Delma J. G. R., Reza A. M., Turney R. D., FPGA Implementation of a Nonlinear Two Dimensional Fuzzy Filter, IEEE International Conference On Acoustics, Speech, and Signal Processing, , Vol 4, pp 2143-2146, 1999.

[7] Delma J. G. R., Reza A. M., Turney R. D; FPGA Implementation of Adaptif Temproral Kalman Filter for Real Video Filtering, IEEE International Conference On Acoustics, Speech, and Signal Processing, Vol 4, pp. 2231-2234, 1999.

[8] Yamada M., Nishihara A., High-Speed FIR Digital Filter with CSD Coefficients Implemented on FPGA, IEICE Trans. Fundamentals, E84-A, 8, pp.1997-2003, 2001.

[9] "Hardware Description Language." Wikepedia: The Free Enyclopedia, http://en.wikipedia.org/wiki/Hardware_description_language

[10] Xilinx Inc., "Virtex-II Pro™ Platform FPGAs: Functional Description," DS083-2(v3.0), December 10, 2003.