İleri Teknoloji Bilimleri Dergisi
Cilt 2, Sayı 2, 30-39, 2013

Journal of Advanced Technology Sciences
Vol 2, No 2, 30-39, 2013

DİZE EŞLEŞTİRME BAŞARIMINI İYİLEŞTİRMEK İÇİN FPGA TABANLI BİR DONANIM MODÜLÜ TASARIMI

Süleyman ÇAKICI[1]*, İbrahim ŞAHİN[2]

[1] Düzce Üniversitesi, Teknoloji Fakültesi, Bilgisayar Mühendisliği Bölümü, 81620, Düzce, TÜRKİYE,
[2] Düzce Üniversitesi, Teknik Eğitim Fakültesi, Elektronik ve Bilgisayar Eğitimi Bölümü, 81620, Düzce, TÜRKİYE

Özet- Dize eşleştirme işlemleri, güncel uygulamalarda oldukça yaygın olarak kullanılmaktadır ve her zaman bilgisayar bilimleri alanında araştırmalara bir odak noktası olmuştur. Yazılım tabanlı olarak gerçeklenen dize eşleştirme işlemi özellikle büyük boyutlu dizelerde eşleme yapıldığında uzun zaman almaktadır. Bu çalışmada, eşleştirme işlemlerini hızlandırarak, dize eşleştirme başarımını iyileştirmek amacıyla FPGA tabanlı bir donanım modülü tasarlanmıştır. Tasarlanan modül, test verileri kullanılarak test edilmiş ve başarım tespiti yapılmıştır. Modülün veri işleme hızı değişik bilgisayarlarla karşılaştırılmıştır. Karşılaştırma sonuçları, tasarlanan modülün, dize eşleştirme işlemlerini bilgisayarlara nispeten yaklaşık 24 kat kadar daha hızlı gerçekleştirilebileceğini göstermiştir.

Anahtar Kelimeler- FPGA, Dize eşleştirme, Donanım Modülü.

DESIGN OF AN FPGA-BASED HARDWARE MODULE FOR IMPROVING THE PERFORMANCE OF STRING MATCHING

Abstract- String matching is a widely used operation in many modern day applications and is always a research focus in the area of computer science. Software-based string matching implementations experience poor performance especially when large arrays are considered. In this study, a hardware module for FPGA chips was designed to speed-up the string matching process. The module was tested using test data and the performance of it was measured. Its performance was compared to some general purpose computers. The results showed that the string matching operations performed using the module can be carried out up to twenty-four times faster than the software implementation running on different PCs.

Key Words- FPGA, String matching, Hardware module.

1. GİRİŞ (INTRODUCTION)

Dize eşleştirme (String matching), bilgisayar bilimleri alanındaki araştırmalar için her zaman bir odak noktası olmuştur. Bir ana metin içerisinde, belirli bir dizenin (desen dize) bir veya birden fazla karşılığının bulunması amacıyla yapılan karşılaştırma işlemleri biçimde basitçe

* suleymancakici@duzce.edu.tr

tanımlanabilir. Arama motoru, veri sıkıştırma, ağ saldırı tespit, bilgisayar virüs imzası eşleştirme ve DNA seri eşleştirme sistemleri gibi güncel birçok uygulamada oldukça yaygın olarak kullanılmaktadır [1]. Söz konusu uygulamalar için sunulan çözümler genellikle yazılım tabanlı olarak gerçekleştirilmektedir. Bununla birlikte, dize eşleştirme işlemleri yazılım tabanlı olarak yapıldığında, CPU zamanı açısından düşük başarım sonuçları ile karşılaşılmaktadır.

Son yıllarda, gerçek zamanlı ve yüksek başarım gerektiren veri işleme uygulamalarının gerçeklemesinde Alan Programlanabilir Kapı Dizileri (FPGA) kullanımının arttığı görülmektedir. Başlangıçta sayısal tasarımların test edilmesi amacıyla geliştirilen FPGA çipleri, FPGA teknolojisindeki güncel gelişmelere paralel olarak, yüksek dereceli paralel çalışabilme kabiliyetine sahip olmuştur [2, 3]. Bu çalışmada, eşleştirme işlemlerini hızlandırarak, dize eşleştirme başarımını iyileştirmek amacıyla FPGA tabanlı bir donanım modülü tasarımı yapılmıştır.

Dize eşleştirme işlemlerini hızlandırmak amacıyla, literatürde sunulan birçok çalışma bulunmaktadır. Boyer ve diğ. ve Knuth ve diğ., büyük bir metin dizesi içindeki belirli dizeleri bulmak için verimli alt-arama algoritmaları geliştirmişlerdir [4, 5]. Ancak bu yaklaşımlar, dize sayıları arttığında iyi bir ölçeklendirme sağlayamamaktadır. Aho-Corasick algoritması, söz konusu dize seti eşleştirme sorununa ilişkin ölçeklenebilir bir çözüm sunmuştur [6]. Aho-Corasick algoritması (ACA), çeşitli dize eşleştirme işlemlerinde yaygın olarak kullanılan bir yöntemdir ve farklı görevler için FPGA donanım uygulamaları geliştirilmiştir. Fide ve Jenks, saldırı tespit ve ağ yönlendirici uygulanmaları üzerine dize eşleme teknikleri ve donanım uygulamaları ile ilgili kapsamlı bir araştırma sunmuştur [7]. Sidhu ve Prasanna dize eşleştirme işlemleri için, bir sonlu durum makinesi (FSM) oluşturma tekniği önermişlerdir [8]. Fakat önerdikleri FSM'ler tümüyle FPGA üzerinde uygulanması gerektiğinden, bu teknik, büyük dize kümelerinde arama yapabilmek için büyük kapasiteli FPGA'ların kullanılmasını gerektirmektedir. Ağ saldırı tespit sistemleri, saldırı modellerini temsil etmek için düzenli ifadeler (regular expressions) kullanır. Artan saldırı sayısına paralel olarak düzenli ifade sayısı da artmaktadır. Lin ve diğ., FPGA üzerindeki alan kullanımını iyileştirmek için, bir düzenli ifadenin birden fazla kullanımını indirgeyen yeni bir paylaşım mimarisi önermiştir [9]. Canella ve Filippo, BioWall için Needleman-Wunshdize eşleştirme algoritmasını hızlandırmak amacıyla FPGA üzerinde paralel çalışan bir uygulama geliştirmişlerdir [10, 11].

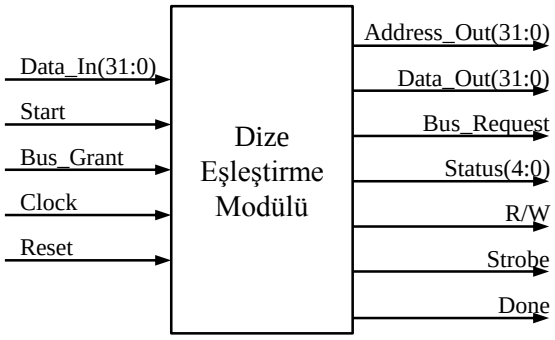
Bu makalede sunulan çalışmada; dize eşleştirme işlemlerini hızlandırmak amacıyla, yukarıda ifade edilen çözümlere alternatif olarak, FPGA çipleri üzerinde çalışabilecek bir donanım modülü tasarlanmıştır. Böylece daha ucuz, daha hızlı ve FPGA çiplerinin defalarca programlanabilme özellikleri sayesinde daha esnek bir yapının ortaya koyulması hedeflenmiştir. Tasarlanan modül, test veriler üzerinde işlemler yapılarak test edilmiş ve modülün ürettiği sonuçların doğrulaması yapılmıştır. Aynı veriler, değişik özellikteki genel amaçlı bilgisayarlar üzerinde de işlenmiştir. Elde edilen sonuçlar kullanılarak, modülün veri işleme hızı bilgisayarlarla karşılaştırılmıştır.

Makalenin ikinci bölümünde, tasarlanan modül detaylarıyla anlatılmıştır. Üçüncü bölümde ise, gerçekleştirilen test çalışmaları ve elde edilen sonuçlar verilmiştir. Elde edilen sonuçlar son bölümde değerlendirilmiştir.

2. DİZE EŞLEŞTİRME İÇİN FPGA MODÜL TASARIMI (DESING OF AN FPGA MODULE FOR STRING MATCHING)

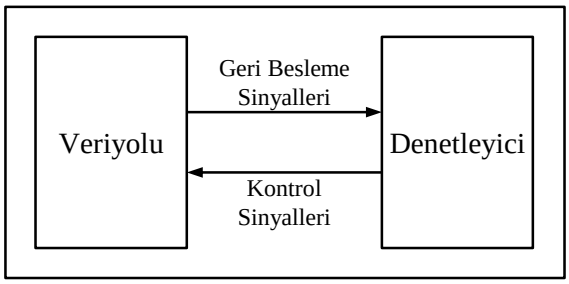
Bu çalışmada, FPGA tabanlı ortamlarda kullanılabilecek bir dize eşleştirme modülü tasarlanmıştır. Tasarlanan modülün en üst seviye blok diyagramı Şekil 1'de gösterilmiştir.

Modül, 32 bitlik `Data_In` girişi ile birer bitlik `Start`, `Bus_Grant`, `Clock`, `Reset` girişlerine sahiptir. Ayrıca, 32 bitlik `Address_Out` ile `Data_Out` çıkışları, 5 bitlik `Status` çıkışı ile birer bitlik `Bus_Request`, `R/W`, `Strobe` ve `Done` çıkışları bulunmaktadır. `Reset` ve `Başla` sinyalleri modülün zamanlaması ve bağlı oldukları bilgisayar arasındaki zaman eşlemesini sağlar. `Bus_Grant` ve `Bus_Request` sinyalleri ise, hafıza ile olan zaman eşlemesini sağlamak ve veri okuma yazma işlemleri için kullanılmaktadır. Tasarlanan modül, bir donanım tanımlama dili olan VHDL'de kodlanmış ve Xilinx ISE 10.1 yazılımı kullanılarak değişik FPGA çiplerine göre sentezlenmiştir.



Şekil 1. Önerilen dize eşleştirme modülünün blok diyagramı (The block diagram of the proposed string matching module)

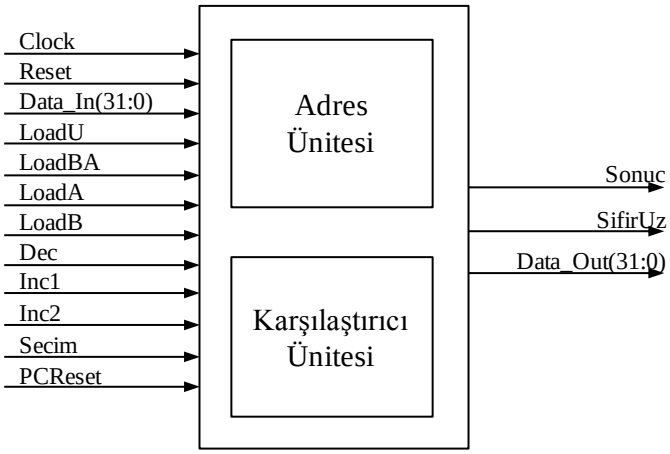
Dize eşleştirme modülünün ikinci seviye blok diyagramı Şekil 2'de gösterilmiştir. Alt-modül, *Veriyolu* ve *Denetleyici* olmak üzere iki blok halinde tasarlanmıştır. *Veriyolu* bloğunda, hafızaya erişim için gerekli adres ünitesi ve hafızadan alınan bilgilerin işlendiği karşılaştırıcı ünitesi bulunmaktadır. *Denetleyici* bloğunda ise, yukarıda bahsedilen işlemlerin zamanını, sırasını, yapılacağı süreyi bildiren ve *Veriyolu*'nu denetleyen sinyaller üretilmektedir.



Şekil 2. Dize eşleştirme modülünün ikinci seviye blok diyagramı (The second-level block diagram of the string matching module)

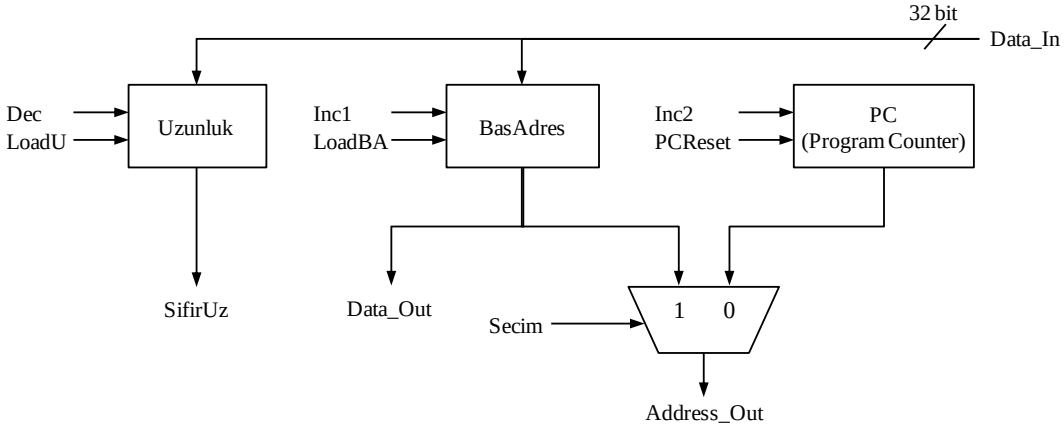
2.1. Veriyolu Bloğu (The DataPath Block)

Veriyolu bloğunda, hafızaya erişim için gerekli adres hesaplamaları yapılmaktadır. Adres ve Karşılaştırıcı olarak adlandırılan iki alt üniteden oluşmaktadır. `Clock`, `Reset`, `Data_in(31:0)`, `LoadU`, `LoadBA`, `LoadA`, `LoadB`, `Dec`, `Inc1`, `Inc2`, `PCReset` ve `Secim` giriş sinyalleri ile `Sonuc`, `SifirUz`, `Data_Out(31:0)`çıkış sinyalleri bulunmaktadır. Şekil 3'te *Veriyolu*'nun blok diyagram görülmektedir.



Şekil 3. Veriyolu blok diyagramı (The block diagram of the DataPath)

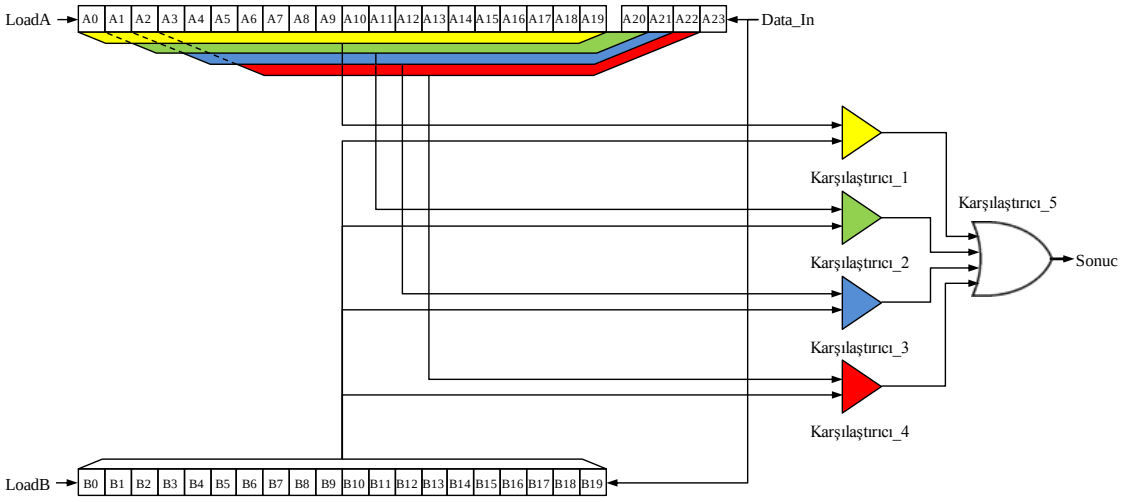
Adres ünitesi *kaydedici* ve *2x1 çoklayıcı* ünitelerinden oluşmaktadır. Şekil 4'te Adres ünitesinin detaylı blok diyagramı görülmektedir. *BasAdres* sayacı hafızadaki içinde arama yapılacak dizenin başlangıç adresini tutan kaydedicidir. *Uzunluk* sayacı ise, arama yapılacak dizenin boyutunu saklamaktadır. `LoadU` ve `LoadBA` sinyalleri, kaydedicilerin içine bilgi yazmak için kullanılmaktadır. `Inc1` ve `Inc2` sinyalleri adres sayaçları içerisindeki bilgileri "1" arttırmak; `Dec` sinyali ise, "1" azaltmak için kullanılmaktadır. *PC* sayacı, başlangıç adresinin, hafızadaki bilgilerin uzunluğunun ve arama yapılacak metinin okunduğu sırada ve son olarak bulunan verinin adresinin 0000_h adresine yazılması sırasında hafıza adreslerini üretmek amacıyla kullanılmıştır. Başlangıç değerleri okunurken, `Inc2` sinyali ile *PC* içerisindeki bilgi "1" arttırılır. Söz konusu okuma işleminin tamamlanmasının ardından *PC* kaydedicisi sıfırlanır. Hafızadan veri okuma esnasında, `Dec` sinyali ile *Uzunluk* sayacının içeriği azaltılırken, `Inc1` sinyali ile de *BasAdres* sayacının içeriği arttırılır. Çoklayıcı ünitesi ise `Secim` sinyaline bağlı olarak *PC* ya da *BasAdres* içeriğini çıkışa aktarmaktadır.



Şekil 4. Adres Ünitesi'nin detaylı blok diyagramı (The detailed block diagram of the Address Unit)

Karşılaştırıcı ünitesi; 8 bitlik 44 adet kaydedici (A0 - A23, B0 - B19), 4 adet 20 baytlık ve 1 adet 4 bitlik karşılaştırma ünitelerinden oluşmaktadır. Hafızadan alınan, arama yapılacak dizeye ait veriler A kaydedicilerine aktarılır. B kaydedicileri ise, aranacak veriyi tutmada kullanılır. A ve B içerisindeki veriler 20 baytlık karşılaştırıcılarda (4 adet) karşılaştırılır. Bu dört karşılaştırma işleminin sonucunda, karşılaştırıcıların herhangi birinin çıkışı mantıksal "1" ise, `Sonuc` sinyali (`Sonuc=1`) üretilir. Şekil 5'te karşılaştırıcı ünitesine ait detaylı blok diyagram

gösterilmiştir. Dört adet farklı karşılaştırıcının kullanılmasının sebebi arama işlemin sırasında içinde arama yapılan dizenin aynı anda ardışıl 4 konumuna birden bakılabilmesi içindir. B kaydedici zinciri 20 bayt ile sınırlandırılmıştır dolayısıyla arama yapılacak kelime en fazla 20 karakter uzunluğunda olabilir. 20 karakterden kısa kelimelerde B kaydedicisinde boş yerlere 00_h ile doldurulur. Karşılaştırıcılar eğer B dizesinde herhangi bir konumda 00_h görürse bu konumları özel olarak değerlendirmekte ve karşılaştırma dışında tutmaktadırlar.

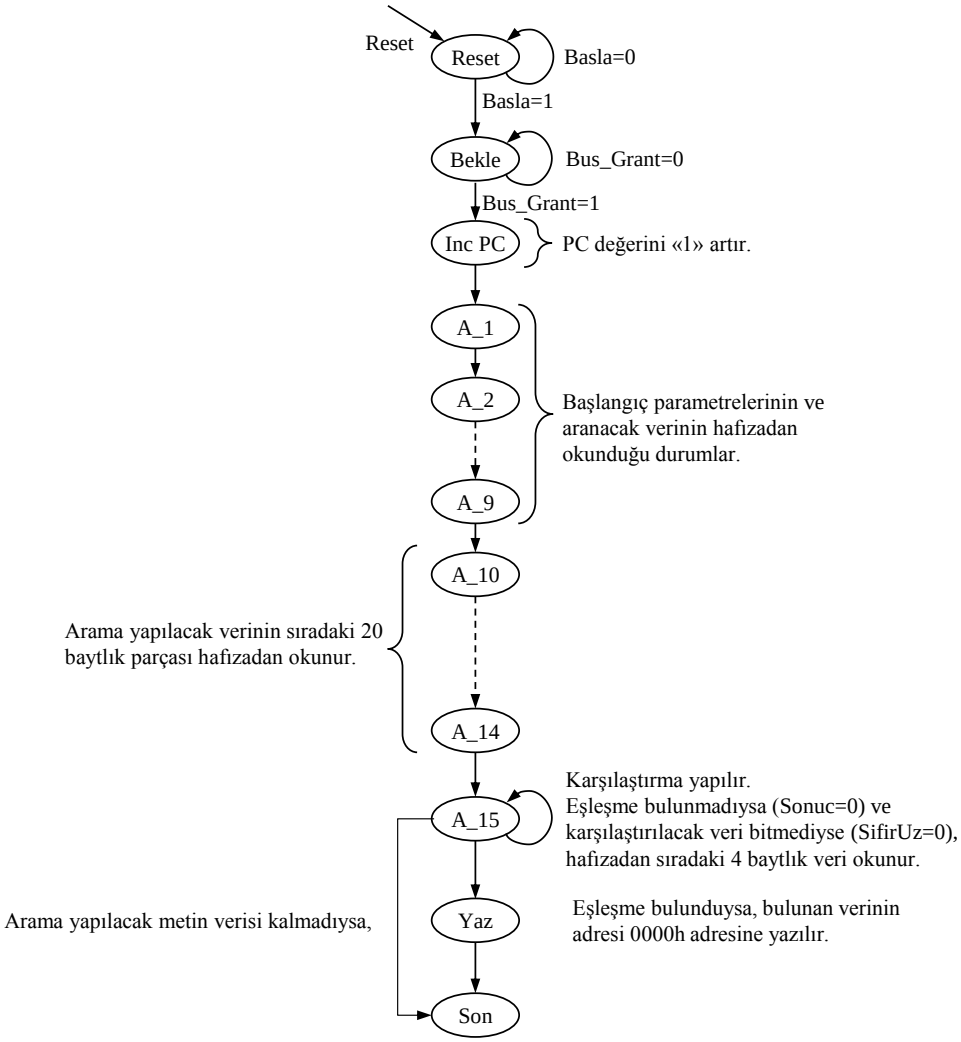


Şekil 5. Karşılaştırıcı ünitesinin detaylı blok diyagramı (The detailed block diagram of the Comparator Unit)

Kaydediciler birbirlerine seri olarak, bir zincir oluşturacak şekilde bağlanmıştır. Bu şekildeki bir bağlantı, kayıtçıların daha kolay kontrol edilmesini sağlamakta; ayrıca, hafızadan okunan bilgilerin kayıcılara aktarılmasında da herhangi bir dezavantaj oluşturmamaktadır. Kaydedici zincirlerine gelen `LoadA` ve `LoadB` sinyalleri, bu iki kaydedici zincirine verilerin düzenli bir şekilde okunmasını ve güncellenmesini kontrol etmektedirler. Her `Load` sinyali alındığında, kaydediciler tarafından tutulan mevcut veriler zincirde bir birim ilerletilmekte ve `Data_In` üzerinden gelen yeni veri ise, zincirde oluşan boşluğa yerleştirilmektedir.

2.2. Denetleyici Bloğu (The Controller Block)

Dize eşleştirme modülü ve modül denetleyicisi 32-bitlik (4 bayt) kelime uzunluğuna sahip hafıza ünitesi ile çalışacak şekilde tasarlanmıştır. Dolayısıyla her hafıza gözünde 4 bayt yani 4 karakter olduğu ve bir seferde 4 karakterin birden okunabildiği göz önünde bulundurulmuştur. Denetleyici toplam 44 durumdan oluşan bir Sonlu Durum Makinesi (Finite State Machine-FSM) olarak tasarlanmıştır. Denetleyicinin durum diyagramı Şekil 6'da gösterilmiştir. Modül ilk çalışmaya başladığında *Reset* durumunda beklenir. `Basla` sinyali alındığında, *Reset* durumundan çıkılarak hafızaya erişmek için sistem yoluna istek sinyali gönderilir. Yolun tahsis edilmesi, yani `Bus_Grant` sinyali beklenir. Yol tahsis işlemi tamamlandığında; başlangıçta 0000_h olan PC değeri "1" artırılarak, 0001_h adresi elde edilir. Denetleyici hafızadaki 0001_h adresine giderek; öncelikle başlangıç adresini, arama yapılacak metin verisinin uzunluk değerini ve aranacak kelimeyi okur ve ilgili kaydedicilere yükler. Bu işlemler 9 durum boyunca tamamlanır. Başlangıç işlemlerinin tamamlanmasının ardından, A_10…A_14 durumlarında arama yapılacak metin verileri hafızadan B kaydedici zincirine okunur. Bu işlemden sonra denetleyici A_15 durumunda bir döngüye girerek, her seferinde hafızadan 4 bayt okur. Okunan verileri A kaydedici zincirine uygun şekilde aktarır; A ve B kaydedici zincirlerindeki bilgilerin karşılaştırma sonucunu kontrol eder.

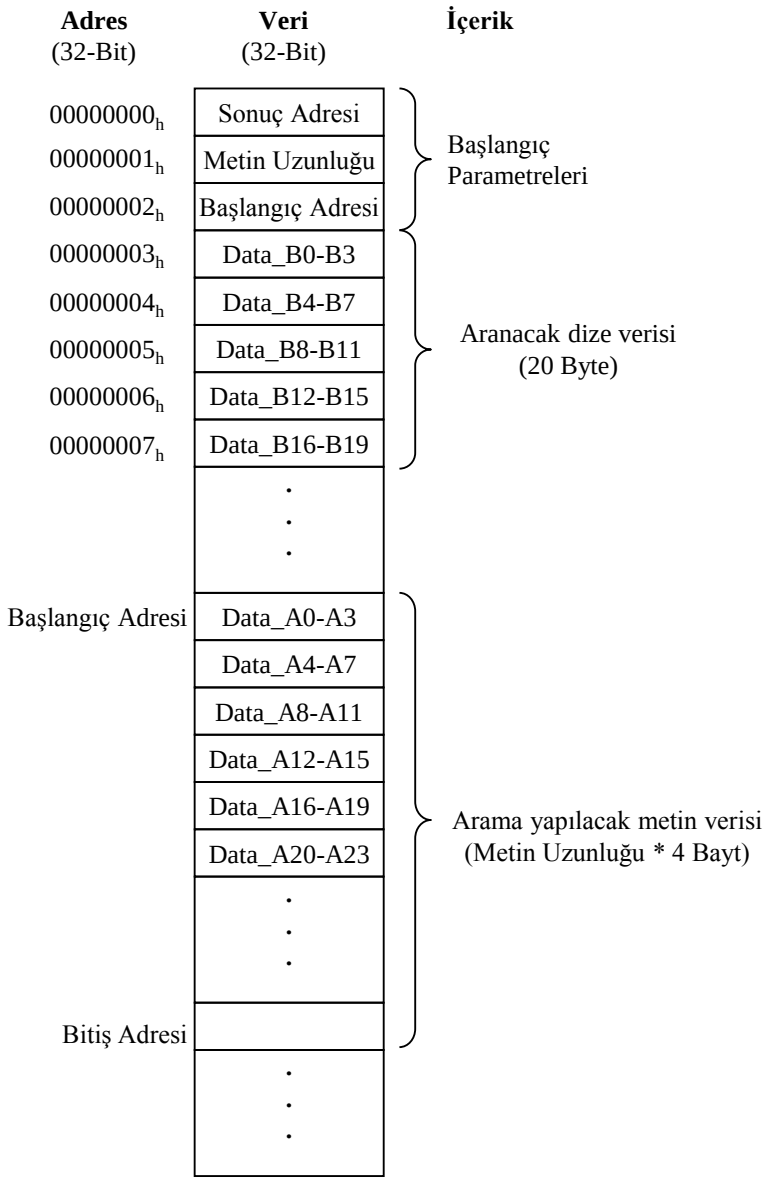


Şekil 6. Dize eşleştirme modülüne ait denetleyicinin durum diyagramı (The controller state diagram of the proposed string matching module)

Karşılaştırma işlemi sonucunda aranan değer bulunduysa "Yaz" durumuna geçilir. PC kaydedicisi sıfırlanır ve aranan kelimenin dize içinde bulunduğu konum hafızada 0000_h adresine yazılır. Hafızadaki arama yapılacak veri bitmediyse ve aranan değer bulunmadıysa, A_15 durumu tekrarlanır. Sıradaki 4 baytlık veri okunduğunda, yine BasAdres değeri "1" arttırılırken, uzunluk değeri "1" azaltılır. Hafızadaki arama yapılacak veri bittiyse, "Son" durumuna gidilir ve işlem sonlandırılır. Denetleyici "Son" durumunda işlemin bitiğini bildirmek için "Bitti" sinyalini gönderir.

2.3. Hafıza Haritası (The Memory Map)

Şekil 7'de tasarlanan modülün kullandığı hafıza haritası görülmektedir. Hafızada, adreslenebilir her bir veri alanının 32-bit olduğu varsayılmıştır. Hafıza haritası iki bölümden oluşmaktadır. Birincisi başlangıç değerlerinin; ikincisi ise, hafızada arama yapılacak metin verisinin kayıtlı olduğu bölümdür. İlk bölüm, hafıza haritasının 0000_h adresinden başlayarak 0007_h adresine kadar olan ilk sekiz adresten oluşur. 0000_h adresi, elde edilen sonucun hafızadaki adresinin saklanması için kullanılır. 0001_h adresinde hafızadaki verilerin uzunluğu, 0002_h adresinde hafızadaki verinin başlangıç adresi, 0003_h-0007_h arasında kalan kısımda ise 20 baytlık aranacak ifade bulunmaktadır.



Şekil 7. Dize eşleştirme modülüne ait hafıza haritası (The memory map of the proposed string matching module)

3. TASARLANAN DİZE EŞLEŞTİRME MODÜLÜNÜN KARŞILAŞTIRMALI TEST SONUÇLARI (THE COMPARATIVE TEST RESULTS OF DESIGNED STRING MATCHING MODULE)

Bu çalışmada tasarlanan dize eşleştirme modülü, Xilinx firması tarafından üretilen *Virtex5* ve *Spartan2E* çipleri için sentezlenerek, modülün FPGA çip istatistikleri ve maksimum saat frekansları incelenmiştir. Modülün belirli uzunlukta ve nitelikteki bir veriyi işleme süresi, ISE benzetim programı kullanılarak elde edilmiş ve farklı PC'lerin aynı veriyi işleme süreleri ile karşılaştırılmıştır. Tablo 1'de modüle ait sentezleme sonuçları görülmektedir. Sentezleme sonuçları göstermiştir ki, kullanılan slice register, LUTs (Look-Up Tables) sayılarına göre, teorik olarak, modülün *Spartan2E* çipine bir, *Virtex5* çipine elli adet kopyasının aynı anda sığması mümkündür. Fakat kullanılan IOB (giriş/çıkış pinleri) dikkate alındığında, her iki çipe de yalnız bir adet modülün rahat bir şekilde yerleştirilebileceği görülmektedir.

Tablo 1. Sentezlenen dize eşleştirme modülünün FPGA çip istatistikleri (The FPGA chip statistics of synthesized string matching module)

FPGA Çip Türü	Slice Reg. Sayısı / %	LUTs Sayısı / %	Slice FFs Sayısı / %	Bounded IOBs Sayısı / %
Virtex 5	453 / 02	523 / 02	453 / 02	108 / 49
Spartan2E	554 / 72	709 / 46	468 / 30	107 / 60

Modülün çalışma performansını karşılaştırmak için özellikleri Tablo 2'de verilen iki adet bilgisayar seçilmiştir. Seçilen bu bilgisayarlardan PC-1 tek çekirdekli, PC-2 çift çekirdekli ve Intel markalı işlemciye sahiptir.

Tablo 2. Deneylerde kullanılan bilgisayarların özellikleri (The features of the computers used in the tests)

PC Adı	CPU Hızı (GHz)	CPU Ön Bellek (KB)	RAM Hafıza Boyutu (MB)	Hafıza Tipi	CPU Türü	FSB BUS Hızı (MHz)	BUS Boyutu (Bit)
PC-1	1.86	2048	1536	DDR2	Intel Centrino	533	32
PC-2	2.00	4096	1536	DDR2	Intel Centrino Duo	666	32

Deneylerde kullanılan 500, 1000, 5000, 10000 bayt boyutlarındaki test dizeleri, bir C program parçacığı kullanılarak oluşturulmuştur. Bilgisayarların, oluşturulan bu test dizelerinde arama başarımının ölçülmesi için, yine C dilinde geliştirilen bir uygulama kullanılmıştır ve elde edilen işleme süreleri (µs) türünden Tablo 3'te verilmiştir. Ölçümler yapılırken, aranan dizenin hedef dize içinde hem ortada hem de sonda olduğu durumlara bakılmıştır. Bu sonuçlar bilgisayarların arama işleminde harcadığı sürenin, hedef dizenin boyutuyla doğrusal olarak değiştiğini göstermektedir. Tablo 4'te ise *Virtex5* ve *Spartan2E* çiplerinin test dizelerini işleme süreleri verilmiştir. Burada kullanılan test dizeleri, aranan dize son kısımda yer alacak şekilde organize edilmiştir.

Tablo 3. PC'lerin farklı boyutlardaki metin verilerini işleme süreleri (PCs' processing times of different sized text data)

Veri Boyutu (Bayt)	PC-1 (*µs*)	PC-2 (*µs*)
500	7.16	4.42
1000	12.90	7.14
5000	58.60	30.41
10000	108.55	59.34

(a) Aranan dize ortada iken

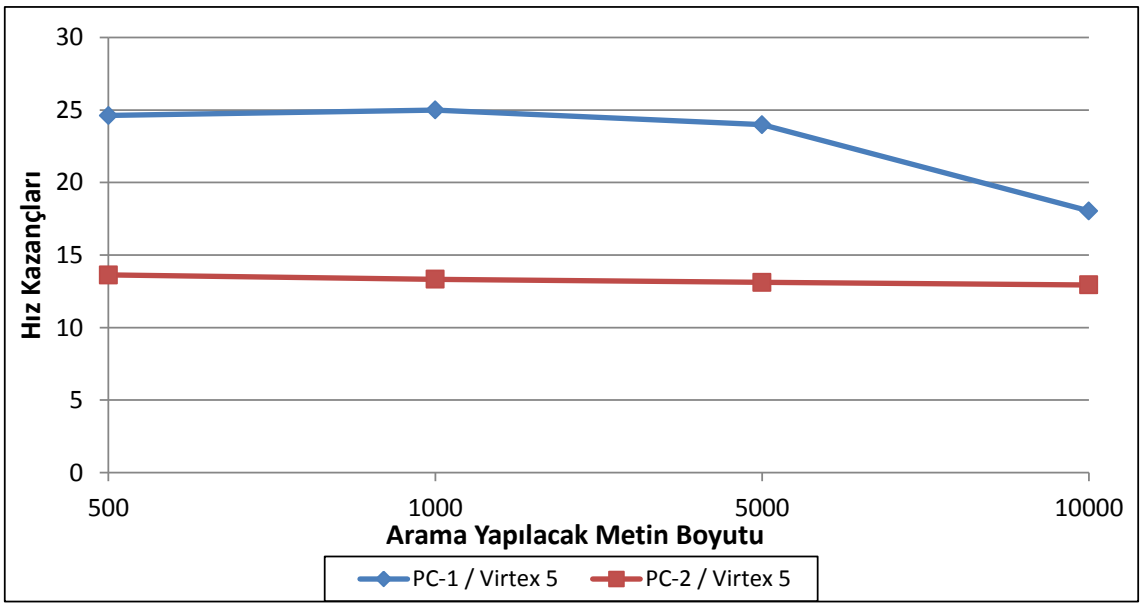
Veri Boyutu (Bayt)	PC-1 (*µs*)	PC-2 (*µs*)
500	12.90	7.14
1000	24.40	13.01
5000	108.55	59.34
10000	164.50	118.08

(b) Aranan dize sonda iken

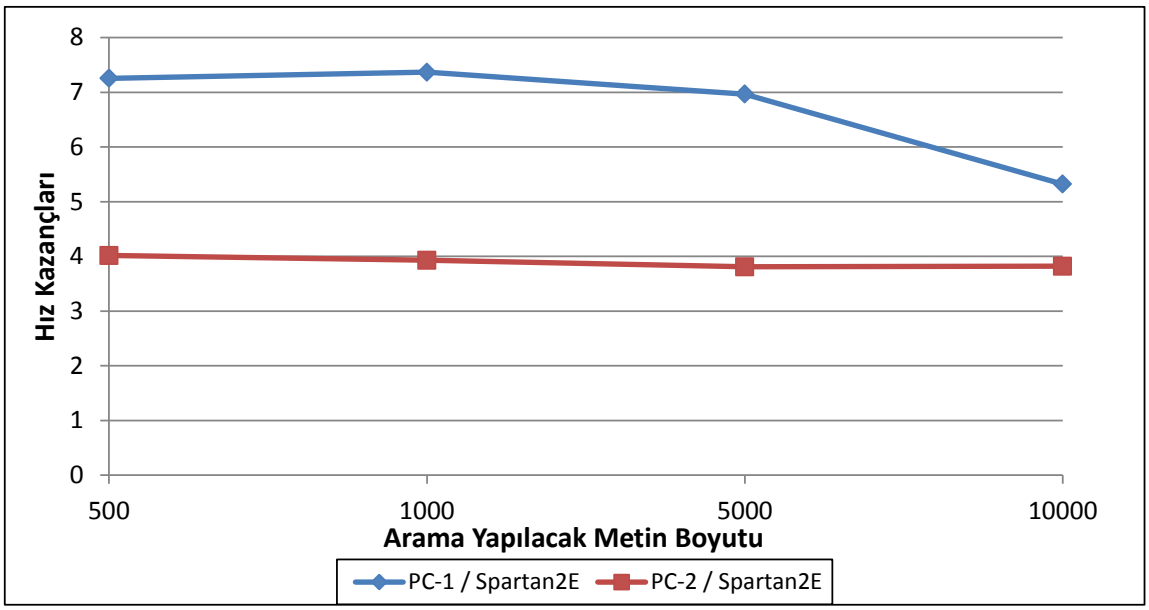
Tablo 4. Seçilen FPGA çiplerinin farklı boyutlardaki metin verilerini işleme süreleri (FPGA chips' processing times of different sized text data)

Veri Boyutu	*Virtex5* (*µs*)	*Spartan2E* (*µs*)
500	0.524	1.778
1000	0.976	3.312
5000	4.525	15.580
10000	9.119	30.915

Şekil 8 ve Şekil 9'da, sırasıyla modülün *Virtex5* ve *Spartan2E* çipleri üzerindeki çalışma sürelerinin PC sonuçları ile karşılaştırılması görülmektedir. Her bir grafikte, farklı boyutlardaki metin verileri için, bir FPGA çipine göre sentezlenmiş modülün deneyde kullanılan bilgisayarlara göre hız kazancı görülmektedir. *Virtex5*'te modül, metin boyutuna göre değişmekle birlikte, PC-1'e göre yaklaşık 24 ve PC-2'ye göre yaklaşık 13 kata kadar daha hızlıdır. *Spartan2E*'de ise; PC-1'e göre 7 ve PC-2'ye göre yaklaşık 4 kata kadar hız kazancı sağlanmıştır. Grafiklerde görülen hız kazançları tek bir çip üzerinde, tek bir modül konfigürasyonu çalıştırıldığında elde edilen sonuçlardır. Birden fazla çip üzerinde birden fazla modül yapılandırması paralel olarak çalıştırıldığında, bu hız kazançlarının katlanarak artacağı gayet açık bir şekilde görülebilmektedir.



Şekil 8. *Virtex5* çipine göre sentezlenmiş modülün PC'lerle başarım karşılaştırılması
(The performance comparison of the module synthesized for *Virtex5* chip and the PCs)



Şekil 9. *Spartan2E* çipine göre sentezlenmiş modülün PC'lerle başarım karşılaştırılması
(The performance comparison of the module synthesized for *Spartan2E* chip and the PCs)

4. SONUÇ VE TARTIŞMA (CONCULUSION AND DISCUSSION)

Dize eşleştirme işlemleri, her zaman bilgisayar bilimleri alanında araştırmalara bir odak noktası olmuştur ve güncel uygulamalarda oldukça yaygın olarak kullanılmaktadır. Bununla birlikte, söz konusu işlemler yazılım tabanlı olarak yapıldığında, CPU zamanı açısından düşük başarım sonuçları ile karşılaşılmaktadır. Bu çalışmada, dize eşleştirme işlemini hızlandırmak amacıyla, FPGA çipleri üzerinde çalışabilecek bir donanım modülü tasarlanmıştır. Tasarlanan modül, test verileri üzerinde işlemler yapılarak test edilmiş ve modülün başarımı genel amaçlı bilgisayarlarla karşılaştırmalı olarak incelenmiştir. Karşılaştırma sonuçlarına göre, tasarlanan modül kullanılarak gerçekleştirilen dize eşleştirme işlemlerinin yaklaşık 7 ile 24 kat arasında hızlandırılabileceği tespit edilmiştir. Hız kazançları tek bir FPGA çipi üzerinde tek bir modülün çalıştırıldığı varsayılarak hesaplanmıştır. Bir FPGA çipine birden fazla modülün uygun şekilde yerleştirilmesiyle veya birden fazla FPGA çipinde modülün birden fazla kopyasının çalıştırılmasıyla hız kazancının katlanarak artacağı düşünülmektedir.

5. KAYNAKLAR (REFERENCES)

[1]. Wang W.; Wu S., (2009). A jumping string mode matching algorithm, 4th International Conference on Computer Science & Education, 1181-1185.

[2]. Dehon, A., (2000). The Density Advantage of Reconfigurable Computing, *IEEE Computer*, 33, 41-49.

[3]. Qasim, S.M., Abbasi S.A., and Almashary, B., (2009). An Overview of Advanced FPGA Architectures for Optimized Hardware Realization of Computation Intensive Algorithms, *IMPACT'09*, 300-303.

[4]. Boyer R.S., Moore J.S., (1977). A Fast String Searching Algorithm, Communications of the ACM, 20, 762-772.

[5]. Knuth D.E., Morris J.H., Pratt V.B., (1977). Fast pattern matching in strings, SIAM Journal of Computing, 6, 323-350.

[6]. Aho A., Corasick, M., (1975). Efficient string matching: an aid to bibliographic search, Communications of the ACM, 18, 333-340.

[7]. Sidhu R., Prasanna, V.K., (2001). Fast Regular Expression Matching Using FPGAs, 227-238.

[8]. Fide S., Jenks, S., (2006). A Survey of String Matching Approaches in Hardware. University of California Irvine.

[9]. Lin, C., Huang, C., Jiang, C., Chang S., (2007). Optimization of Pattern Matching Circuits for Regular Expression on FPGA, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 15(12), 1303-1310.

[10]. Canella, M.; Miglioli, F., (2003). Performing DNA comparison on a bio-inspired tissue of FPGAs, International Parallel and Distributed Processing Symposium, 1-8.

[11]. Tempesti, G., Mange, D., Stauffer, A., Teuscher, C., (2002). The BioWall: An electronic tissue for prototyping bio-inspired systems, IEEE proceedings of NASA/DoD Conference on Evolvable Hardware, 221–230.