KOCAELİ ÜNİVERSİTESİ * FEN BİLİMLERİ ENSTİTÜSÜ

FPGA TABANLI DOKUNMATİK EKRANLI KULLANICI ARABİRİM TASARLANMASI VE GERÇEKLEŞTİRİLMESİ

YÜKSEK LİSANS

Fatih AKKOYUN

Anabilim Dalı: Elektronik ve Bilgisayar Eğitimi

Danışmanı: Prof. Dr. Kadir ERKAN

KOCAELİ, 2011

KOCAELİ ÜNİVERSİTESİ * FEN BİLİMLERİ ENSTİTÜSÜ

FPGA TABANLI DOKUNMATİK EKRANLI KULLANICI ARABİRİM TASARIMI VE GERÇEKLEŞTİRİLMESİ

YÜKSEK LİSANS TEZİ

Fatih AKKOYUN

Tezin Enstitüye Verildiği Tarih: 05 OCAK 2011

Tezin Savunulduğu Tarih: 11 ŞUBAT 2011

Tez Danışmanı
Prof.Dr. Kadir ERKAN
(.....)

Üye
Doç.Dr. Mehmet YILDIRIM
(.....)

Üye
Yrd.Doç.Dr. Seda POSTALCIOĞLU ÖZGEN
(.....)

KOCAELİ, 2011

ÖNSÖZ VE TEŞEKKÜR

Gömülü sistemlerde insan makine arabirimlerinin önemi gün geçtikçe artmaktadır. İnsanlar hep daha kolay kullanılan, daha çok geribildirim sunabilen arabirimleri talep etmektedirler. Ancak bu arabirimlerin gerçekleştirilmesi gömülü sistemin gerçekleştirilmesinin yarısı kadar zaman alabilmektedir. Bu tez çalışmasında, arabirim tasarımıda yaşanan zaman kayıplarını azaltmak hedeflenmiştir.

Proje ve tez aşamasında bana yol gösteren Sn. Prof. Dr. Kadir ERKAN'a teşekkür ederim.

İÇİNDEKİLER

ÖNSÖZ VE TEŞEKKÜR i
İÇİNDEKİLER ii
ŞEKİLLER DİZİNİ iv
TABLOLAR DİZİNİ v
KISALTMALAR DİZİNİ vi
Özet. vii
Abstract viii
BÖLÜM 1. GİRİŞ 1
BÖLÜM 2. FPGA TABANLI DOKUNMATİK EKRANLI ARABİRİMLER 5
BÖLÜM 3. SİSTEM MİMARİSİ 9
3.1. Bilgisayar Ortamı 11
3.2. FPGA Geliştirme Kartı 11
3.3. Dokunmatik Panelli LCD Ekran 12
BÖLÜM 4. DONANIM 13
4.1. FPGA Teknolojisi 13
4.2. NexysII FPGA Geliştirme Kartı ve Özellikleri 14
4.3. Dokunmatik Ekran Teknolojileri 15
4.3.1. Dirençli teknoloji 16
4.3.2. Kapasitif 19
4.3.3. Yüzey akustik dalgası 19
4.3.4. Kızıl ötesi 19
4.3.5. Gerinim ölçme 20
4.3.6. Optik görüntüleme 20
4.4. ELT240320TP Dokunmatik Panelli LCD Ekran ve Özellikleri 21
4.4.1. ELT240320TP bağlantı şeması 21
4.4.2. ELT240320TP blok biyagramı 22
4.4.3. ELT240320TP uç yerleşimi 23
4.4.4. ELT240320TP uç giriş çıkışları 23
4.5. FPGA Kartı ile LCD Ekran Bağlantıları 25
BÖLÜM 5. YAZILIM 26
5.1. IP Çekirdeği Oluşturma 26
5.2. Seri Haberleşme Birimi IP Çekirdeği 29
5.2.1. SPI birimi özellikleri: 31
5.2.2. SPI birimi sistem blok diyagramı 31
5.2.3. SPI birimi blok diyagramı 32
5.2.4. Fonksiyonel tanımlamalar 33
5.2.4.1. Sinyal tanımlamaları 33
5.2.4.2. Saat üretimi 33
5.2.4.3. Temel işlem 35
5.2.4.4. SPI saatli kaydırma biçimleri 35
5.2.4.5. SPI kesmeler 35
5.2.5. SPI yazmaç bellek haritası 35
5.2.6. SPI yazmaçları ve kontrol bitleri 37

5.2.6.1. Gönderilen veri yazmacı (SPI_TX_DATA) 37
5.2.6.2. Alınan veri yazmacı (SPI_RX_DATA) 37
5.2.6.3. Saat denetleme (SPI_CLK_CTRL) 37
5.2.6.4. Kurulum yazmacı (SPI_SETUP) 38
5.2.6.5. Durum yazmacı (SPI_STATUS) 40
5.2.7. Zamanlama bilgileri 40
5.3. LCD Ekran IP Çekirdeği 41
5.4. MicroBlaze Tabanlı C Kodları 43
5.4.1. Ana program (Main.c) 43
5.4.2. Dokunmatik panel fonksiyonları (TC240320TP.c) 44
5.4.3. LCD Ekran fonksiyonları (LCD240320TP.c) 44
BÖLÜM 6. UYGULAMA 46
6.1. Hiper Terminal Bağlantı Ayarları 46
6.2. Ekran Tasarımında Kullanılan Yazılım 47
6.3. LCD Ekran Fonksiyonlarının Çalışma Şekli 49
6.4. LCD Örnek Ekran Fonksiyonları ve Tanımlamaları 49
6.5. Örnek Uygulamalar 51
6.5.1. Uygulama 1 51
6.5.2. Uygulama 2 52
6.5.3. Uygulama 3 53
6.5.4. Uygulama 4 54
BÖLÜM 7. SONUÇLAR VE ÖNERİLER 56
7.1. Sonuçlar ve Öneriler 56
KAYNAKLAR 58
EKLER 60
ÖZGEÇMİŞ 103

ŞEKİLLER DİZİNİ

Şekil 2.1: Tipik bir mikrodenetleyici temelli ekran mimarisi 7
Şekil 2.2: FPGA temelli ekran mimarisi 8
Şekil 3.1: Gerçekleştirilen Sistemin Blok Diyagramı 10
Şekil 4.1: Digilent Nexys2 (Xilinx Spartan 3E-500 FG320) FPGA blok diyagramı 14
Şekil 4.2: Dirençli Dokunmatik Ekran Yapısı 17
Şekil 4.3: 2.8" TFT Renkli LCD 240 x 320 piksel dokunmatik ekran 21
Şekil 4.4: ELT240320TP Dokunmatik LCD Ekranın PCB kartındaki uç bağlantıları [25]. 22
Şekil 4.5: ELT240320TP Blok Diyagramı [5]. 22
Şekil 4.6: ELT240320TP Dokunmatik LCD Ekranın Uç Yerleşimi 23
Şekil 4.7: ELT240320TP Dokunmatik LCD Ekranın Uç Tanımlamaları 25
Şekil 5.1: Veri yoluna bağlanmış bir çevresel birimin temel yapısı 28
Şekil 5.2: Çevresel birimdeki kullanıcı mantığı hiyerarşisi 29
Şekil 5.3: SPI Sistemi Blok Diyagramı [21]. 31
Şekil 5.4: Üst Düzey SPI Blok Diyagramı [21]. 32
Şekil 5.5: SPI Master MISO örneklemesi [21]. 40
Şekil 5.6: SPI IP Çekirdeği zamanlama diyagramı [21]. 41
Şekil 5.7: ELT240320TP Normal Yazma Modu Zaman Çizelgesi [5]. 42
Şekil 6.1: Terminal Bağlantı Noktası Ayarları 46
Şekil 6.2: Ekran tasarımında kullanılan yazılım 48
Şekil 6.3: Terminal ekranına girilen fonksiyonlar 51
Şekil 6.4: Fonksiyonların LCD ekranda işlenmiş hali 51
Şekil 6.5: Terminal ekranına girilen fonksiyonlar 52
Şekil 6.6: Fonksiyon LCD ekranda işlenmiş hali 52
Şekil 6.7: Ekran tasarımnın yapıldığı yazılım 53
Şekil 6.8: Fonksiyon LCD ekranda işlenmiş hali 53
Şekil 6.9: Ekran tasarımnın yapıldığı yazılım 54
Şekil 6.10: Fonksiyon LCD ekranda işlenmiş hali 54

TABLOLAR DİZİNİ

Tablo 4.1: ELT240320TP Dokunmatik LCD Ekranın Uç Bağlantıları [25]. 24
Tablo 5.1: SPI Sinyal adlandırma ve tanımlamaları 33
Tablo 5.2: SPI_SCK_FREQ[2:0]'a göre SPI_SCK Frekans karşılığı 34
Tablo 5.3: SPI_CORE IP Çekirdeği Yazmaç Adresleri..... 36
Tablo 5.4: SPI_SS'ın SPI_SS_SETUP değerine göre davranış durumları [21]. 39
Tablo 5.5: SPI IP Çekirdeği zamanlama sınırları [21]. 41
Tablo 5.6: ELT240320TP Normal Yazma Modu Zaman Sınırları [5]. 42

KISALTMALAR DİZİNİ

AIR	: Adobe Çalışam Ortamı
BSB	: Temel Sistem Oluşturucu
CIP	: Çevresel birim oluşturma veya ekleme
CLB	: Yeniden düzenlenebilir mantık bloğu
CRT	: Katot Işın Tüpü
CS	: Tümdevre seçimi
EDK	: Gömülü Sistem Geliştirme Aracı
FPGA	: Sahada Programlanabilir Kapı Dizisi
GUI	: Grafiksel kullanıcı arayüzü
HCI	: İnsan Bilgisayar Etkileşimi
HDL	: Donanım Tanımlama Dili
HMI	: İnsan Makine Arabirimi
IOB	: Giriş/Çıkış Blokları
IP	: IP Çekirdeği
IPIF	: IP Çekirdeği Arayüzü
İBE	: İnsan Bilgisayar Etkileşimi
LCD	: Sıvı Kristal Ekran
OEM	: Orijinal Ürün Üreticileri
PC	: Kişisel Bilgisayar
RS232	: Seri iletişim portu
SCK	: Saat
SDI	: Seri veri girişi
SDO	: Seri veri çıkışı
SPI	: Çevresel Seri Arabirim
VHDL	: Çok Yüksek Hızlı Donanım Tanımlama Dili
XPS	: Xilinx Platform Stüdyo

FPGA TABANLI DOKUNMATİK EKRANLI KULLANICI ARABİRİM TASARLANMASI VE GERÇEKLEŞTİRİLMESİ

Fatih AKKOYUN

Anahtar Kelimeler: İnsan Makine Arabirimi, FPGA, Dokunmatik LCD, Gömülü Sistemler, MicroBlaze, VHDL, IP(IP Çekirdeği)

Özet: Günümüzde savunma, sayısal işaret işleme, uzay, tıbbi görüntüleme ve otomotiv alanlarında Sıvı Kristal Ekran (LCD) kullanımı yaygındır. Kullanıcı ile etkileşim içeren bu cihazlarda sadece gelişmiş arabirimler yardımıyla etkin iletişim sağlamak mümkündür.

Bu tezde Sahada Programlanabilir Kapı Dizisi (FPGA) tabanlı bir gömülü sistemde dokunmatik LCD Ekranların kullanılması anlatılmaktadır. Burada FPGA tabanlı kullanıcı arabiriminin düşük maliyet ve yüksek performans gözetilerek, hızlı bir şekilde tasarlanması ve gerçekleştirilmesi hedeflenmiştir. Ek olarak sabit işlemcilerle geliştirilen aygıtlar yerine, farklı sistemlere uygulanabilen farklı ekranların kullanılabildiği bir sistem oluşturma hedeflenmiştir.

Oluşturulan sistemde, insan makine etkileşimini sağlayan dokunmatik ekranlı kullanıcı arayüzünün tasarlanması seri port yoluyla sağlanmıştır. Ekran tasarımı bilgisayardan gönderilen komutların gömülü sistemde işlenerek dokunmatik ekranda görüntülenmesi ile gerçekleştirilmektedir. Bu sayede tasarımcı yazılımın sentezlenmesi ve yüklenmesi aşamalarında zaman kaybı yaşamamaktadır ve tamamen arabirimin tasarımına yoğunlaşabilmektedir

Tezde kullanıcı arabirimi olarak dokunmatik panelli LCD ekran ve geliştirme ortamı olarak FPGA kullanıldı. Kullanıcı arabirimin yapısal tasarımında Nexys2 geliştirme kartı kullanılmıştır. Gerçekleştirilen insan makine arabiriminin yeniden kullanılabilir olması ve farklı geliştirme kartlarına kolay kullanımı için sistem IP Çekirdeklerinden (Intellectual Property-Fikri Mülkiyet) oluşturulmuştur.

Gerçekleştirilen bu sistem, tasarım aşamasında gerçek cihazlar kullanılarak sentezleme ve yükleme adımları olmadan, doğrudan arabirimin tasarlanılabilmesi amacına temel teşkil edecek biçimde oluşturulmuştur.

THE DESIGN AND IMPLEMENTATION OF FPGA BASED HUMAN MACHINE INTERFACE WITH TOUCH SCREEN

Fatih AKKOYUN

Keywords: Human Machine Interface, FPGA, Touch Screen LCD, Embedded Systems, MicroBlaze, VHDL, IP(Intellectual Property)

Abstract: Nowadays, Liquid Crystal Display (LCD) is common in military defence, digital signal processing, aerospace, medical imaging, and automotive areas. Interaction with the user in these devices it is only possible to obtain effective communication with advanced interfaces

In this thesis, the usage of touch LCD Screens in a FPGA based embedded system is described. In this way, it is aimed to design and implement FPGA based user interfaces quickly with the understanding of low cost-high performance theory. In addition, instead of the devices developed with fixed processors, it is aimed to design a system in which different screens can be used and also which can be applied to different systems.

In this designed system, the design of touch screen user interface which allows interaction between human and machine is enabled via serial port. The screen design is achieved after the commands, sent from the computer, are processed in embedded system and displayed in touch screen. By this means, in the process of synthesis and installation of designer software no time is wasted, and so it is possible to focus only on the design of the interface.

In this thesis, touch panelled LCD screen, as user interface, and FPGA, as progressing context, are used. In the design of user interface the Nexys 2 Development Card is used to reduce the amount of cost.

This system is designed by using real devices in the process of design without the synthesis and installation steps which enables to design the interface directly.

BÖLÜM 1. GİRİŞ

Grafik LCD ekranlar, İnsan Makine Arabirimi (HMI) olarak endüstriyel konsollar, otomatik satış makineleri, otomotiv ve deniz sektörü, ev aletleri, tıbbi cihazlar ve oyun makineleri gibi gömülü sistemlerin görsel denetiminde gittikçe yaygınlaşmaktadır. Verimli bir denetleyici tasarımında etkili bir çözüm için hem maliyet azaltılması hem de performans artırılması zorunludur [1]. Tüketici aygıtlarında ve benzeri elektronik aygıtların üretiminde düşük maliyetli LCD'lere yönelim artmaktadır. İnsan Bilgisayar Etkileşimini (İBE) önemli ölçüde etkileyen Lcd'ler, ev uygulamalarında geleneksel "mekanik" insan makine arabirimlerinin yerlerini ele geçirmiş durumdalar.

Ekranlar, özellikle grafiksel HMI'lar günümüzde, insan makine etkileşimi içeren sayısız uygulama ile kolay etkileşimi arayan tüketiciler için önemi gittikçe artan bir rol oynamaktadırlar. Tüketiciler bu aygıtlardan "akıllı" olmalarını, bilgileri kolayca ve verimli bir şekilde iletebilmelerini beklemektedirler [2]. Tasarımcılar genellikle tüketicilerin bu isteğine hızlı ve etkili şekilde cevap verebilen, kendi tasarımlarıyla kolayca bütünleştirilebilen kararlı yapıdaki üçüncü parti ekran denetim çözümlerine ihtiyaç duymaktadır [1].

Sistem tasarımcıları tasarımlarında üst düzey görsel çözümler gerçekleştirebilmek için FPGA gibi programlanabilir mantık devrelere yönelmişlerdir. Bu sayede tasarımlarında çoklu ekranları ve işlemcileri destekleyerek değişen market standartlarına ve gereksinimlerine uyum sağlayabilmektedir [1].

Bu özelliklerin tümü bir tek noktada birleşmektedir: insan makine arabirimi etkileşimi için üst düzey ekranlara ihtiyaç duyulmaktadır. Çoğu durumda bunu başarmak için dokunmatik panelli Lcd'lere ihtiyaç duyulmakta, orijinal ürün üreticilerine (OEM) modern üretim imkânı sağlamakta, sık uygulama kontrol imkânı sunmakta ve kendilerini diğer markaların sınıflarına çıkararak müşteriyi memnun etmektedirler. Dijital ekran teknolojileri mekanik HMI'ların olduğu bir yerde

kullanımı nihayetinde daha hızlı ürün geliştirmeye yol açar. Bileşenlerine ayrıştırılmış mimari sayesinde yeniden kullanıma imkân tanır. Sonuç olarak Grafiksel Kullanıcı Arayüzü (GUI) yazılımına ve FPGA'lara bağlı olarak daha ekonomik tasarımların gerçekleştirilmesine imkân tanır [2].

Çalışmanın temel amacı gömülü sistemlerde kullanılan insan makine arabirimlerinin bir gömülü sisteme bağlı olmaksızın tüm sistemlerle kolayca bütünleştirilebilen, arayüz tasarımı açısından oldukça hızlı olan ve düşük maliyetle herhangi bir FPGA kartında kullanılabilen kullanıcı arabirimi gerçekleştirmektir.

Bu tezde insan makine etkileşimini sağlayan dokunmatik ekran kullanıcı arayüzünün tasarlanması, bilgisayardan gönderilen komutların sentezleme ve derleme aşaması olmadan doğrudan dokunmatik ekranda çalıştırılması ile gerçekleştirilmektedir. Bu sayede tasarımcı yazılımın sentezlenmesi ve yüklenmesi aşamalarında zaman kaybetmeden attığı her adımı LCD ekranda görebilmektedir.

İlk aşamada LCD FPGA haberleşmesine temel olacak olan haberleşme birimleri gerçekleştirildi.

İkinci aşamada LCD temel fonksiyonlarının FPGA üzerinde C ile oluşturulan fonksiyonlar sayesinde LCD ekranda görüntülenmesi gerçekleştirildi.

Üçüncü aşamada hiper terminalden gelen grafik kodları FPGA ortamında işlenerek LCD ekranda görüntülenmesi test edildi

Son aşama olarak Adobe Air tabanlı geliştirilen bir tasarım aracı ile oluşturulan LCD fonksiyonlarının seri port yoluyla gönderilerek LCD ekranda çalıştırılması gerçekleştirildi.

Bu çalışmada Digilent firmasının ürettiği 500K kapılı Nexys2 FPGA geliştirme kartı kullanıldı. Kullanıcı arabirimi olarak ELT240320TP adlı dokunmatik panelli LCD kullanıldı. Kullanıcı arabirimi için, öncelikle RAM' a yazılan veriler TCLCD_CORE

tarafından işlenerek LCD ekrana gönderildi. Dokunmatik panelden alınan veriler SPI_CORE IP Çekirdeği ile alınarak veriler kullanılabilir hale getirildi. Bu sayede kullanıcı makine ile arasında başka analog bir aracı bulunmadan dokunmatik panelli ekrandan etkileşim yapabilir hale geldi. Temelde insan makine arabirimi için LCD ekran için çıkış birimi, dokunmatik panel ise giriş birimi olarak kullanıldı.

Çalışmada kullanılan IP çekirdekleri özgün olarak tasarlandı ve gerçekleştirildi. Kullanılan IP çekirdekleri sayesinde oluşturulan HMI istenilen gömülü sisteme kolaylıkla bütünleştirilebilmektedir.

Bölüm 2'de gömülü sistemlerde kullanılan İnsan Makine Arabirimleri(HMI), Dokunmatik Panelli LCD'lerin gömülü sistemlerde kullanımı, FPGA tabanlı insan makine arabirimleri ile ilgili son yıllardaki gelişmeler gözden geçirildi.

Bölüm 3'te Tasarımda gerçek cihaz kullanımı için oluşturulmuş sistemin blok diyagramı verilerek sistem mimarisi anlatıldı.

Bölüm 4'te Sistem bileşenleri ve bu bileşenlerin kullandığı teknolojilerden bahsedildi.

Bölüm 5'te Xilinx XPS (Xilinx Platform Studio) kullanarak VHDL tabanlı IP (Intellectuel Property - IP Çekirdeği) oluşturulması ve özgün olarak oluşturulan Dokunmatik Ekran IP Çekirdeği ile Seri Haberleşme IP çekirdeği çalışma şekli ve kullanılması anlatıldı. Bunun yanında sistemde kullanılan MicroBlaze tabanlı c kodları ve işlevleri anlatıldı.

Bölüm 6'da Arabirim tasarımı için gereken hiper terminal bağlantı ayarları, arabirim tasarımı için hazırlanış örnek fonksiyonların çalışma şekli, örnek fonksiyonların listesi ve bu fonksiyonlarla yapılmış uygulamalarla birlikte Adobe Air platformunda hazırlanmış tasarım aracı ile LCD ekran tasarlanması anlatıldı.

Bölüm 7'de bu tez çalışmasının sonuçları, bilim dünyasına kazandıracakları değerlendirildi.

Bu değerlendirmeler ışığında, bu konuda gelecekte neler yapılabileceğine ilişkin düşüncelerden bahsedildi.

BÖLÜM 2. FPGA TABANLI DOKUNMATİK EKRANLI ARABİRİMLER

Geleneksel yaklaşımlarda insan makine arabirimleri olarak butonlar ve ledlerden oluşan mekanik arabirimler kullanılmaktaydı. Ancak günümüzde LCD ekranların maliyetlerinin azalması ve yüksek etkileşimli grafik arayüzleri sayesinde geleneksel olan "mekanik" arabirimler yerlerini LCD ekranlara bırakmaktadır [3]. Grafik ekranların popülerliği ev otomasyonlarında, taşınabilir bilgisayarlarda, cep telefonları gibi yaygın olarak kullanılan cihazlarda artmasının yanında endüstriyel uygulamalarda da artış göstermektedir. Talepteki bu artış rekabeti artırarak ekran kalitesinin gelişmesini sağlamakta ve maliyetleri de azaltmaktadır [4].

Ekran teknolojileri gün geçtikçe gelişen çözünürlükleri, artan görünürlük açıları, artan görüntüleme hızları sayesinde marketlerdeki yerlerini korumaya devam edeceklerdir. Ayrıca istenildiğinde LCD ekran üzerine yerleştirilebilen saydam dokunmatik panellerde ekranların marketlerdeki konumlarını korumasına yardımcı olmaktadır [4]. Kullanıcıların cihazlardan daha "akıllı" daha kolay ve etkin bir biçimde bilgiyi iletmesi gibi beklentileri LCD ekranların bu konumunu sabitlemektedir [3].

Günümüzde kullanıcıların cihazlarından beklentilerinin artışı, daha akıllı cihazlara yönelmeleri, piyasada rekabetin artmasını sağlamaktadır. Bu beklentilerin artması ve LCD ekranların hızlı gelişiminin sonucu olarak bir cihazın ömrü 2-3 yıl ile sınırlanmaktadır. Bu ekranlardan gelişime ayak uyduramayanlar, yeterli çözünürlük ve hızı sağlayamayanlar, kullanıcıların ihtiyaçlarını karşılayamamaktadır. Dolayısıyla piyasaya sürülen bir ürünün ömrü 2-3 yıl ile sınırlı kalmaktadır [4].

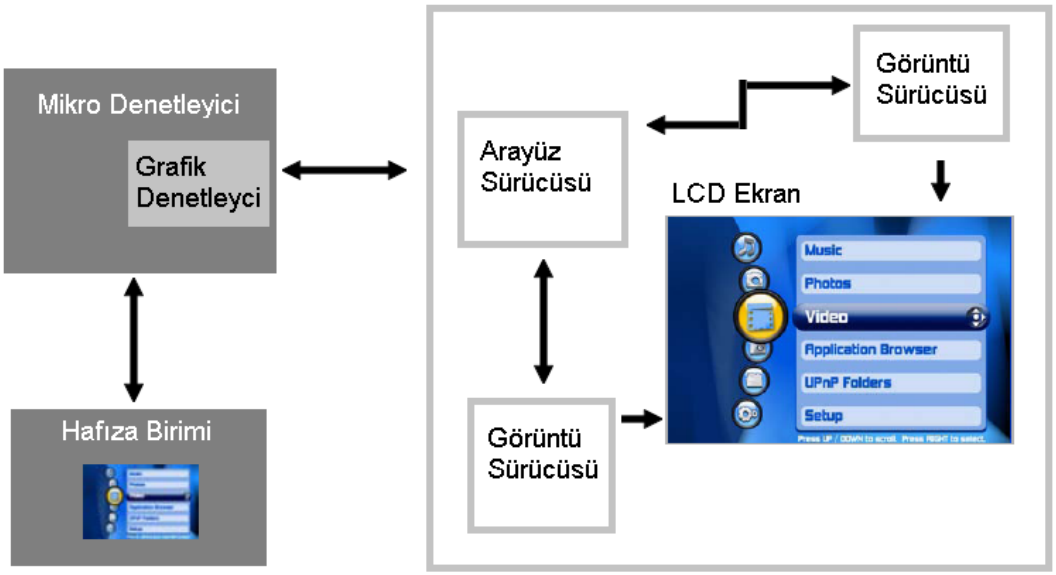
Tüm bu etkenler göz önüne alındığında, hepsinde ortak olan özellik insan makine etkileşimini kolaylaştırmak için yüksek görüntü teknolojisine ihtiyaç duymaktadırlar. Çoğu durumda başarılı bir dokunmatik ekran kullanılması orijinal ürün üreticilerine (OEM) modern ve gösterişli uygulama kontrolü sağlayan bir cihaz üretmelerine

imkân vermektedir. Bunun yanında dijital ekran teknolojilerinin kullanımı hızlı üretime, karmaşık mimarilerin yeniden kullanılabilmesine, bunların sonucu olarak da daha ekonomik arayüz tasarımı yapılabilmesini sağlamaktadır [3].

Günümüzde marketlerde çeşitli özelliklerde LCD sürücüleri ve denetleyicileri bulunmaktadır. Bunlar zekâsı bulunmayan saf görüntü sürücüleri (zekası olmayanlar), hafızalı görüntü denetleyicisi ve genel amaçlı mikrodenetleyici ile bütünleştirilmiş hafızalı ekranlardır. Saf görüntü sürücüleri genellikle, yeterli tazeleme hızına ulaşmak için yüksek frekansta seri giriş ile devamlı yeni görüntü verisi bekler. Görüntü denetleyicili ekranlarda ise denetleyici bir dizi grafik kodu ve denetleme kodunu gönderilmesini ve dâhili hafızasında görüntüyü çoklanmasını kendisi yönetir.

Çoğu yaygın mikrodenetleyicinin bütünleştirilmiş görüntü denetleyicisi bulunmaktadır. Bu aygıtlar kendine özgü giriş çıkışlara sahiptir ve bazılarında yüksek performans için dâhili hafızada bulunmaktadır.

Bir grafik sistemi oluşturmanın en kolay yolu mikrodenetleyici temelli LCD modülü kullanmaktadır. Modülde görüntü sürücü aygıtı bulunmaktadır ve doğrudan mikrodenetleyiciye bağlanmaktadır. Görüntülenecek ya da depolanacak resimler mikrodenetleyicide çalışan bir program sayesinde mikrodenetleyicinin hafızasında saklanabilmektedir. Bu aygıtlar genellikle ekrana özgün grafik motoru ve özel etkili görüntülere sahiptir. Şekil 2.1'de yaygın olarak kullanılan mikrodenetleyici temelli ekran mimarisi görüntülenmiştir [4].

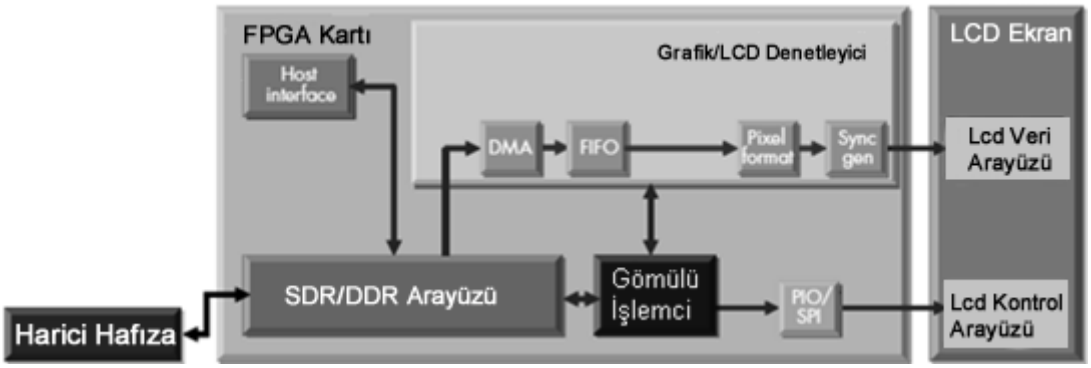


Şekil 2.1: Tipik bir mikrodenetleyici temelli ekran mimarisi

Ancak bu kolaylıklara rağmen, mikrodenetleyici ve grafik motorundaki sabitlenmiş özelliklerden dolayı mikrodenetleyici kullanımı kısıtlanmaktadır. Piyasada birçok denetleyici bulunmasına rağmen bunlardan bazılarında grafik işleme motoru olması seçim şansını da azaltmaktadır. Bunların yanında sisteme yeni bir özellik (giriş/çıkış aygıtı, dokunmatik panel, yeni grafik arayüzü) ihtiyacı olduğunda grafik işleme motorunda değişiklik yapılamamaktadır. Bunları desteklemesi için daha yüksek maliyetli bir aygıt alınması gerekecektir. Bu da zaman kaybına ve maliyet artışına yol açan yeni bir yazılım mimarisi ve yeni port bağlantıları gerektirecektir [4].

Burada mikrodenetleyicilerin sınırlılığını ve problemlerini içermeyen özgün mikroişlemci çekirdekleri tasarlanabilen, hazır IP çekirdekleri sunan ve kendi mantıksal yapımızı tasarlayabildiğimiz FPGA'lara (programlanabilir kapı dizileri) ihtiyaç duyulmaktadır. FPGA tabanlı bir görüntü birimi istenildiğinde mikrodenetleyici kadar basit bir yapıda istenilirse daha karmaşık işlemler gerçekleştirebilen genişletilmiş yapıda tasarlanabilir [4].

Şekil 2.2'de FPGA temelli ekran mimarisi gösterilmektedir.



Şekil 2.2: FPGA temelli ekran mimarisi

FPGA yapısının esnekliği ve mikrodenetleyicilerin sınırlılıkları göz önüne alınarak sistem oluşturulurken dokunmatik ekranlı arabirim FPGA tabanlı olarak gerçekleştirilmiştir. Sonraki bölümde gerçekleştirilen sistemin blok diyagramı verilerek sistemin mimari yapısı anlatılacaktır.

BÖLÜM 3. SİSTEM MİMARİSİ

Kullanıcıların, kullandıkları makinelerden, giderek artan görsellik ve kullanışlılık beklentisi sonucu yaygınlaşan dokunmatik ekranlı arabirimlerin, gömülü sistemlerde kullanımı günümüzde oldukça yaygınlaşmıştır. Kullanıcılara kolaylık sağlayan arabirimler, bu kolaylıkların yanında sistem tasarımcısının tasarım süresini büyük ölçüde artırmaktadır. Farklı işlevleri olan gömülü sistemlerde, insan makine arabirimleri, sisteme özgün olarak tasarlanması gerekmektedir. Bundan dolayı arabirimler, gömülü sistemi gerçekleştirme süresini etkileyen önemli bir etkendir.

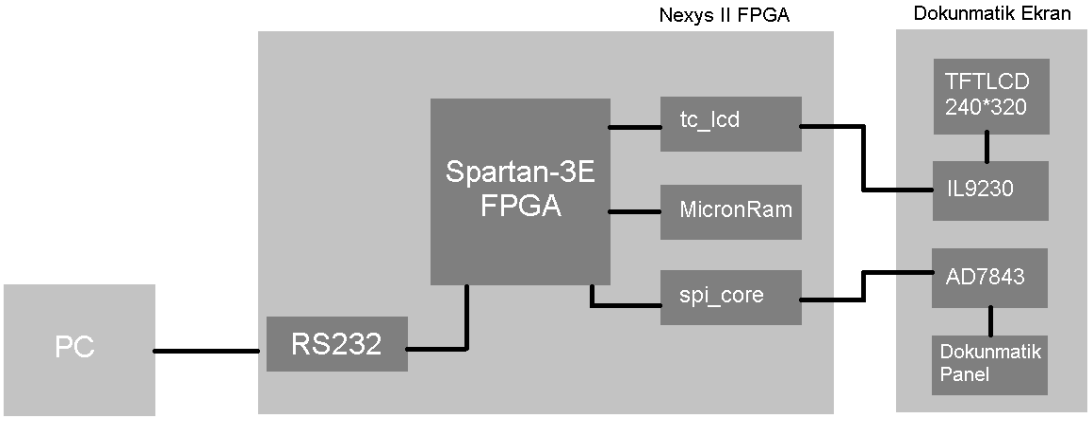
Bu tezde insan makine arabirimi tasarımı aşamasında yaşanan zaman kayıplarını azaltabilmek hedeflendi. Bu hedef doğrultusunda tezde HMI tasarımı için dokunmatik panelli LCD ekran (ELT240320TP) kullanıldı [5]. Geliştirme kartı olarak Digilent NEXYS2 FPGA kartı seçildi [6]. Ayrıca gerçekleştirilen dokunmatik arayüzün farklı gömülü sistemlerde kullanılabilmesi için özgün IP Çekirdeği (IP) yazılarak sistemle bütünleştirildi. Seçilen FPGA kartında bulunan RS232 seri port bağlantısı kullanılarak LCD ekran arayüz tasarımı, bilgisayardan terminal aracılığıyla gönderilen komutlarla sağlandı.

Bu sistemin mimarisi gömülü sistemlerde arabirim tasarımının önemi göz önüne alınarak hazırlanmıştır. Buradaki mimari gömülü sistemlerde arabirim tasarımını hızlandırmak ve kolaylaştırmak amacıyla geliştirilmiştir.

Sistem tasarımında, mikroişlemci olarak Xilinx firması tarafından geliştirilmiş olan 32-bitlik, RISC mimarisine sahip MicroBlaze işlemcisi kullanılmıştır. MicroBlaze işlemci FPGA içerisindeki blok RAM'leri desteklemektedir ve FPGA için geliştirilmiş olan bir mikroişlemcidir [7].

Oluşturulan sistemde MicroBlaze için çalışma frekansı ile işlemci-veri yolu frekansı 50 MHz seçilmiştir. Veri ve program belleği olarak kullanılacak olan BRAM boyutu 16 KB olarak belirlenmiştir. Grafik ekran tasarımında kullanılan ilkel grafik

fonksiyonlarını seri port üzerinden FPGA kartına gönderebilmek için standart giriş ve çıkış olarak RS232 seçilmiş ve RS232 biriminin hızı 115200 olarak belirlenmiştir. Önyükleme belleği olarak BRAM seçilerek sistem başlatıldığında burada yer alan yazılımın çalışması sağlanmıştır.



Şekil 3.1: Gerçekleştirilen Sistemin Blok Diyagramı

Bu sistemde RS232 yoluyla gelen LCD fonksiyonları MicroBlaze işlemcisi tarafından FPGA kartı üzerinde işlenerek çevresel birimler yardımıyla Dokunmatik panelli LCD ekrana gönderilmektedir. Şekil 3.1'de sistemin genel yapısı gösterilmektedir. Sonraki aşamalarda sistem daha ayrıntılı diyagramlarla anlatılacaktır. Temel olarak Şekil 3.1'de PC'den gelen LCD komutları RS232 yoluyla FPGA'ya gelmektedir. FPGA'da gelen komutlar işlenerek LCD Ekranda gösterilmektedir.

Bu mimaride arabirim olarak kullanılan dokunmatik ekran tasarımı işlemi bilgisayar ortamına aktarılmıştır. Bu sayede tasarımcı arabirim tasarlarken sentezeleme ve yükleme adımlarından kurtularak tasarımını doğrudan LCD Ekran üzerinde görebilecektir. Bu mimaride sistem tasarımcısı tasarlayacağı sistemi bilgisayar ortamında tasarlamakta ve sonuçlarını gömülü sistemin arabirimi olan LCD Ekranda görebilmektedir. Oluşturulan sistemin mimarisi bilgisayar ortamı, FPGA geliştirme kartı, dokunmatik panelli LCD Ekrandan oluşmaktadır. Bu bileşenler sırasıyla açıklanaktır.

3.1. Bilgisayar Ortamı

Gömülü sistemlerde bir arabirim tasarımı gerçekleştirmek ve her defasında tasarımı FPGA'ya yükleyip kontrol etmek oldukça zaman alıcı bir işlemdir. Bu işlemi zamansal açıdan azaltabilmek ve arabirim tasarımını kolaylaştırmak için tasarımda kullanılan grafik fonksiyonlarının bilgisayardan doğrudan FPGA'ya gönderilerek çalıştırılması amaçlanmıştır.

Burada bilgisayar, terminal bağlantısı ve Adobe Air'de oluşturmuş olduğumuz masaüstü yazılım sayesinde, seri porttan FPGA'ya, LCD Ekran fonksiyonlarının gönderilmesinde kullanılmaktadır. Bu sayede tasarımcı sadece bilgisayar kullanarak gömülü sistem arabirimi tasarımını gerçekleştirebilmektedir.

3.2. FPGA Geliştirme Kartı

Gömülü donanımlar işlemci veri yolları sayesinde birbirine bağlanan, bir ya da daha fazla işlemci içeren, çevresel birimlerden ve hafıza bloklarından oluşan yapılardır. Bu donanımlar dış dünya ile portlar yolu ile bağlantı kurabilmektedirler. Bu bağlantılar ve işlemcilerin çokluğu karmaşıklığa yol açsada firmaların sunduğu geliştirme ortamları sayesinde bu tür karmaşıklıklar indirgenebilmektedir. Bu sayede istenilen dış birimler ile iletişim sağlanması oldukça kolaylaşmaktadır. [8].

Bu faydaları içeren FPGA'lar çoğu gömülü sistemde yaygın olarak kullanılmaya başlamıştır. Bunun yanında kullanıcıların ihtiyacı olan sistemleri istenilen esneklikte geliştirmeye imkân tanımaktadırlar. FPGA'lar sayesinde sistem istenirse yeniden şekillendirilebilmekte veya tamamen değiştirilebilmektedir. FPGA'lar paralel işlem yapabilme yeteneği ve yüksek performansının yanında maliyeti de düşüktür. Bu nedenlerden dolayı sistemin mimarisinde temel bileşen olarak FPGA kullanılmıştır.
Sistemde bulunan FPGA sayesinde bilgisayardan gelen LCD Ekran fonksiyonları MicroBlaze ortamında LCD Ekrana uygun olarak işlenmektedir. MicroBlaze'de işlenen bu fonksiyonlar VHDL kullanılarak oluşturulmuş olan LCD Ekran IP çekirdeği üzerinden LCD Ekranda bulunan IL9230 sürücüsüne gönderilmektedir.

3.3. Dokunmatik Panelli LCD Ekran

Dokunmatik Ekranlar günümüzde kullanıcıların yatak odasına kadar girmiştir. Bu ekranlar kullanıcılara yüksek çözünürlükte görsellik sunabilmektedirler. Görsel olarak estetik olmalarının yanında dokunmatik olmalarından dolayı bulundukları ortamlarda fazla yer kaplamamaktadırlar. Bu tür ekranların kullanıcı talepleri doğrultusunda günümüzde yaygın olarak kullanılmaları maliyetlerini de oldukça azaltmıştır. Ayrıca bu ekranlar alt seviyede firmalara orijinal ürün üretebilme imkânı tanımaktadır. Bu avantajlardan yaralanabilmek için sistemde insan makine arabirimi olarak dokunmatik panelli LCD Ekranı kullanmayı tercih ettik.

Bu sistemde seçilen dokunmatik panelli LCD Ekranda bulunan IL9230 gelen renk bilgilerini ve komutları işleyerek TFT LCD ekranda görüntülemektedir. Dokunmatik panel bölümü ise AD7843 analog dijital dönüştürücü sayesinde kullanıcının dokunduğu noktanın X,Y koordinatlarını alarak seri haberleşme ip çekirdeğine iletmektedir. Seri haberleşme ip çekirdeği gelen veriyi işleyerek dokunulan noktayı belirlemektedir.

BÖLÜM 4. DONANIM

4.1. FPGA Teknolojisi

Günümüzde sayısal sistem tasarımlarında çok sık kullanılan FPGA, Üretimden bağımsız programlanabilen mantıksal kapı dizileri olarak tanımlanır. FPGA'lar programlanabilir mantık devre blokları, bu blokları birbirine bağlayan ayarlanabilir ara bağlantılardan oluşan tüm devrelerdir [9] "Sahada" söylemi, programlamanın üreticide elindeyken değil de, müşteride iken yani imalat aşaması sonrasında tasarımcı tarafından yeniden programlanabilmesi sahada yapılmasındandır [10].

FPGA "mantık blokları" olarak adlandırılan programlanabilir mantık bileşenleri içermesi sayesinde, temel lojik kapıları (AND, OR ya da XOR) ya da kod çözücü gibi daha karmaşık yapıdaki devre elemanları olarak yapılandırılabilirler [11].

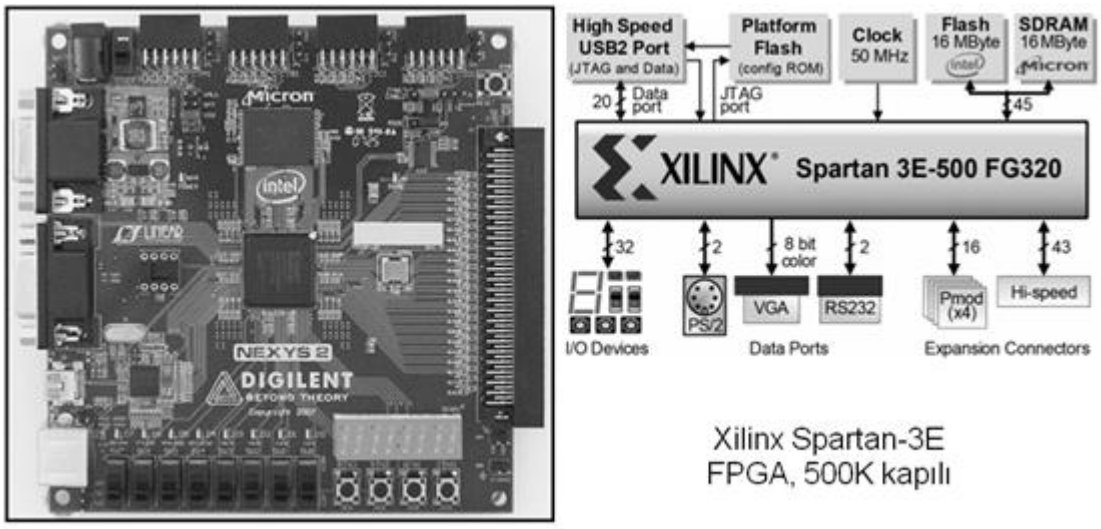
Tasarım sırasında kullanıcıya sağladığı esneklik, düşük maliyet ve hızlı ilk üretme özelliği ile FPGA'lar sayısal tasarım ortamlarının vazgeçilmez yapıları haline gelmiştir. Uygulama alanlarından bazıları sayısal işaret işleme, kriptografi, uzay, savunma ve tıbbi görüntülemedir [12].

FPGA'lar Donanım Tanımlama Dili (HDL) ile tanımlanan kapı dizilerini donanımsal olarak gerçekleştirirler. Bu donanım tanımlama dillerinden VHDL ile Verilog en yaygın kullanılan iki HDL türüdür. Bir FPGA mikroişlemci gibi tasarlanıp mikroişlemcilerin fonksiyonlarını gerçekleştirebilirler. Bunun yanında mikroişlemcilerden farklı olarak paralel çalışan fonksiyonları işleyebilirler. Bu sayede birçok mantık hücresinden oluşan FPGA'nın hücrelerine ayrı ayrı fonksiyonlar atanarak ayrı görevler verilebilir [11,20].
Hazırlanan bu sistemdeki grafikler istenirse herhangi bir işlemci veya mikrodenetleyici kullanılarak gerçekleştirilebilir. FPGA kullanılmasındaki sebep ise özgün IP çekirdekleri oluşturabilme ve bir ekrana ya da yazılıma bağımlı kalmadan

sistemi gerçekleştirebilmektir. Hazırlanan sistemde Digilent firmasının üretmiş olduğu 500K kapılı Xilinx firmasını ürettiği Spartan-3E FPGA'sına sahip Nexys2 geliştirme kartı kullanılmıştır.

4.2. NexysII FPGA Geliştirme Kartı ve Özellikleri

Bu tezde kullanılan Nexys2 FPGA geliştirme kartı ve özellikleri aşağıdadır.
Bu kart Digilent Firması tarafından Xilinx Spartan 3E FPGA kullanılarak üretilmiştir. Kart üzerinde yüksek hızlı USB2 portu, birkaç giriş/çıkış aygıtı ve her türde dijital sistemler, gömülü işlemciler temelli Xilinx MicroBlaze için ideal ortam sunmaktadır [6]. Şekil 4.1'de Nexys2 geliştirme kartının görüntüsü ve blok diyagramı bulunmaktadır.



Şekil 4.1: Digilent Nexys2 (Xilinx Spartan 3E-500 FG320) FPGA blok diyagramı

Nexys2 Geliştirme Kartı Özellikleri

- 500K-kapılı Xilinx Spartan 3E FPGA
- USB2-temelli yüksek-hızlı veri transferi ve FPGA (Adept Suite Software kullanıldığında destekler)
- USB bağlantısı ile çalıştırılabilir (bataryalar ve/veya şebeke gerilimi ilede kullanılabilir)
- 16MB Micron PSDRAM ve16MB Intel StrataFlash ROM

- Xilinx Platform Flash kalıcı FPGA yapılandırmaları için
- Güç kaynağı ile batarya arasında etkin geçiş (batarya kaynaklı uygulamalar için ideal)
- 50MHz osilatör ve ikinci bir osilatör için soket
- 60 FPGA I/O(giriş/çıkış) genişleme konektörleri ile yönetilebilen(yüksek hızda Hirose FX2 konnektörü sayesinde)
- 8 LED, 4-digit 7-seg display, 4 buton, 8 slâyt anahtarı [6].

4.3. Dokunmatik Ekran Teknolojileri

Dokunmatik ekran herhangi bir LCD veya CRT ekran üzerine yerleştirilmiş doğrudan ekran üzerinden giriş alabilen teknolojidir. Bu teknoloji dokunmatik ekran kalemi veya ekran yüzeyine dokunmayla kullanılabilir. Dokunmatik ekranlar basınca duyarlıdır; kullanıcı ekrandaki kelimelere ve yazılara dokunarak bilgisayarla etkileşim sağlar.

Bir dokunmatik panel görüntülenen alanın fiziksel sınırları içinde, el ya da bir cisim aracılığıyla yapılan dokunuşun basıncını ve konumunu belirler. Dokunmatik Ekran üç temel (Dokunma Sensörü, Denetleyici, Sürücü)bileşenden oluşur. Bu ekranlarda en önemli bileşen dokunma algılayıcı sensördür. Bu algılayıcı sayesinde dokunuş algılanarak bir yazılım aracılığı ile bilgisayarın anlayabileceği dile çevrilmektedir [2].

Dokunmatik ekranlar birçok teknoloji kullanılarak geliştirilebilmektedir. Bu teknolojiler aşagıda listelenmiştir:

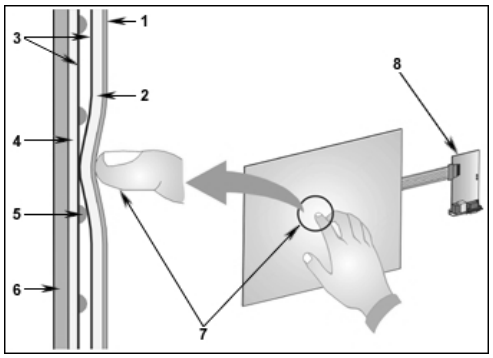
- Dirençli
- Kapasitif
- Yüzey Akustik Dalgası
- Kızılötesi
- Gerinim Ölçme
- Optik Görüntüleme

Günümüzde bu teknolojilerden Dirençli, Kapasitif, Yüzey Dalgası ve Kızılötesi teknolojisi öne çıkmıştır. Bu tezde kullanılan dokunmatik ekran paneli, 4 telli dirençli teknoloji temelli bir dokunmatik paneldir. Bu teknoloji ve çalışma şekli sonraki başlıklarda açıklanacaktır. Ayrıca diğer teknolojilerden kısaca bahsedilecktir.

4.3.1. Dirençli teknoloji

Dirençli dokunmatik ekranlar cam veya plastik alt tabakadan boşluklar bırakılarak ayrılmış ince esnek bir kaplama kullanırlar. Alt tabaka ve üstteki kaplamada saydam metalik şeritler kullanılması sayesinde kullanıcı ekrana basınç uyguladığında katmanlar arasında elektrik geçişi sağlanmış olur. Dokunmanın etkisiyle oluşan elektriksel değişim denetleyici tarafından not alınır ve sürücü yazılımı tarafından bilgisayarın anlayabileceği şekilde güncellenerek fare hareketlerinin aktarılması gibi bilgisayara aktarılır.

Dirençli teknolojiler öncelikle tüm ekranı basınçla çalışan bir anahtarlama sistemi haline dönüştürülür. Bunun için özel bir yapıya sahip kaplama ekran üzerine sıkı bir şekilde yerleştirilir. Bu kaplama iki katmandan oluşur: Üstte dış etkilere dayanıklı polyester panel, altta ise direnç özelliği gösteren panel. Üstteki panelin de ön ve arka yüzeyleri de farklı özelliklere sahiptir. Ön yüzey dış etkilere dayanıklı bir yapı sunarken, arka yüzey ise yarı iletkendir. Dokunma işleminin algılanması için, öncelikle üst kaplamadaki iletken yüzey ve alttaki dirençli kaplamanın bir şekilde birbiriyle temas etmesi gerekir. Ancak bunun bir dokunma etkisiyle olması gerektiğinden, her iki kaplama arasına yerleştirilen yüzlerce şeffaf ayıraç sayesinde paneller arasından bir hava boşluğu oluşturarak iki kaplamanın durup dururken birbiriyle temas etmesi engellenir. Şekil 4.2'de katmanları gösterilen dirençli dokunmatik ekranın yapısı ve çalışması:



Şekil 4.2: Dirençli Dokunmatik Ekran Yapısı

1. Polyester Film (Dış Kaplama)
2. Üstteki dirençli katman
3. İletken ITO (Saydam metal kaplama)
4. Alttaki dirençli katman
5. Ayırıcı noktalar
6. Cam Panel
7. Ekran yüzeyine dokunarak üstteki dirençli katman(2) ile alttaki dirençli katman(4) arasında dokunulan nokta ile devre tamamlanır.
8. Dokunmatik ekran denetleyicisi (7) iki katman arasından birbiri ardına voltaj değerleri alır ve dokunulan noktayı dijital X ve Y koordinatları haline getirir.

Bu işlemler sonucunda ekran büyük bir anahtarlama sistemine dönüşür. Alttaki dirençli kaplama, dört adet tel tarafından sürekli olarak sırayla düşey ve yatay eksenler üzerinde hareket eden +5 volt gerilimle beslenmektedir ve kaplamanın direnç özelliği sayesinde bu voltaj bir taraftan diğer tarafa doğru azalan bir değerle ilerler. Yani dirençli kaplamaya voltajı verdiğinizde bir tarafta +5 volt ile yola çıkan voltaj, diğer tarafa doğru yol alırken giderek azalır ve diğer uçta topraklamayla sonlanır. Ancak X ve Y eksenlerinde dönüşümlü olarak verilen bu elektrik akımının voltajındaki azalma, dirençli kaplamanın özelliği sayesinde ekran üzerinde düzenli bir dağılım oluşturur, örneğin X ekseni üzerinden akım verilirken +2.5 volt ölçüm yaptığınız bir noktanın ekranın X ekseni üzerinde tam olarak nereye denk geldiği konusunda tutarlı bir tahmin yapabilir hale gelirsiniz.

Herhangi bir dokunma olmadığı zaman iletken kaplama üzerindeki voltaj değeri

sıfırdır. Ancak ekranın herhangi bir yerine dokunarak iletken ve dirençli kaplamalar arasındaki teması sağladığınızda, iletken yüzey üzerindeki voltaj değişerek denetleyiciyi harekete geçirir. Daha sonra denetleyici, koordinatları belirlemek için sırayla şu işlemleri gerçekleştirir:

Öncelikle X ekseni üzerinde hareket eden bir elektrik akımı oluşturarak iletken yüzeye bağlı denetleyicide beliren voltaj değerini okunur ve kontrol kartındaki işlemci tarafından X konumu belirlenir.

İkinci olarak aynı işlem bu kez Y ekseni üzerinde hareket eden bir elektrik akımı üzerinde gerçekleştirilir ve Y konumu belirlenir.

X ve Y konumlarına dair elde edilen voltaj ölçümleri denetleyici tarafından ölçülüp yorumlandığında, iletken kaplamanın dirençli kaplamaya hangi noktada değdiği anlaşılır ve bu bilgi sayısal hale çevrilerek ilgili yazılıma gönderilerek, dokunma işlevinin gerçekleşmesi sağlanır.

Bu teknolojinin kullandığı kaplama görüntü kalitesini bir miktar etkilemekle birlikte, yine kaplamanın özelliklerinden ileri gelen ciddi avantajları mevcuttur. Örneğin kaplama olarak kullanılan polyester malzeme, dış ortam koşullarına ve ağır kirlilik şartlarına camdan daha fazla dayanıklılık gösterebilir. Ayrıca üzerine yapışan toz, kir gibi etkenler dokunma etkisi yaratmadığından dolayı ve eldiven, kalem gibi ekrana baskı uygulayabileceğiniz her türlü dokunma etkisiyle çalışabilirler. Bu özelliği nedeniyle açık alan uygulamalarında ve ağır kirlilik koşullarında çalışılması gereken durumlarda; örneğin endüstride, hastanelerde, sürekli yanınızda dere tepe gezdireceğiniz el bilgisayarlarının ekranlarında ve kamuya açık alanlarda kullanılan cihazlarda bu teknolojiden faydalanılır [15-19].

Hazırlanan sistemde 240*320 piksel ekran çözünürlüğünü destekleyen 4 telli dirençli dokunmatik panelli ekran teknolojisini kullanan ThaiEasyElec firmasının üretmiş olduğu ELT240320TP modeli kullanılmıştır. Bu modelin ayrıntıları bir sonraki başlık altında anlatılmaktadır.

4.3.2. Kapasitif

Kapasitif sistemlerde elektrikle yüklenmiş bir katman ekrandaki cam panel üzerine yerleştirilmiştir. Ekrana el ile ya da iletken bir cisimle dokunuldugunda kapasitif katmandaki yükün bir kısmı kullanıcıya aktarılır. Bu yük değişimi ekranın köşelerine yerleştirilmiş olan devreler tarafından ölçülür. Kontrolcü her köşedeki yük değişimlerini hesaplayarak tam olarak dokunulan noktayı algılar ve bu noktayı sürücü yazılımına gönderir. Sürücü yazılımı tarafından bilgisayara anlayabileceği şekilde dokunulan nokta aktarılır.

Kapasitif sistemlerin çalışması için ekranın elektrik akımını alabilecek nesne kullanılması zorunludur. Ayrıca kapasitif teknolojide yüzeyde bulanan capacitive folyo'nun yanlış kimyasallar ile silinmesi duyarlılığın bozulmasına neden olabilmektedir.

Kapasitif sistemler ekran ışığının %90 kadarını geçirirler. Bu özellik bakımından %75' eyakın geçirgenlik sağlayan dirençli sistemlerden daha avantajlı durumdadırlar. Bu geçirgenlik kapasitif sistemlerin daha kaliteli ekran görüntüsüne sahip olmalarını sağlar.

4.3.3. Yüzey akustik dalgası

Yüzey ses dalgalı sistemlerde ekranın bir ucundan diğer bir ucuna ses dalgaları gönderen bir verici ve bu dalgaları alan bir alıcı vardır. Yüzey ses dalgalı ekranların çalışma prensipleri, üretilen ses dalgalarının vericiden alıcıya ulaşmasının (dokunuş ile) engellendiği noktanın (hassas ve kesin bir şekilde) tesbit edilmesine dayanır.

4.3.4. Kızıl ötesi

Kızıl ötesi teknolojisi ekran yüzeyini yatay ve dikey yönlerde tarayan ışık demetlerinden oluşmuştur. X ve Y koordinatlarına dizilmiş infrared ışın yayıcı diyotların karşılarında infrared algılayıcılar bulunmaktadır. Ekrana dokunulma eylemi sırasında, yayılan ışının karşı sırada bulunan algılayıcıya erişimi engellenmiş

olur. Bu engelleme sonucu oluşan değişim kontrol kartı ile algılanarak ilgili sürücüye bildirilir.

Diğer teknolojilere göre ölçümleme gerektirmemesi bir avantaj iken, ekrana çok yaklaşan bir cismin dokunulmadan da algılanması dez avantaj olarak söylenebilir.

4.3.5. Gerinim ölçme

Kuvvet Paneli (force panel) Teknolojisi olarakta adlandırılan teknoloji, ekranda dört köşeye yerleştirilmiş gerilimölçerler sayesinde dokunma gerçekleştiğinde oluşan gerilim sapmasını ölçerek dokunmayı algılayan teknolojidir.

Bu teknoloji 1960' lardan beri bulunmaktadır ancak Vissumo ve F-Origin tarafından geliştirilmesiyle ticari yolları açıldı. Bu tekknoloji ayrıca Z eksenini ve insan dokunuşundaki basıncıda ölçebilmektedir.

Bu tür ekranlar genellikle halk sistemlerinde özellikle bilet makinalarında sert kullanımın olduğu alanlarda kullanılmaktadır

4.3.6. Optik görüntüleme

Dokunmatik ekran teknolojisine göre yeni olan teknolojide, iki ya da daha fazla görüntü sensörü ekranın kenarlarına (genellikle köşelerine) yerleştirilir.
Kızılötesi arka ışıklar ekranın diğer tarafındaki kameranın görüntüleyeceği alana yerleştirilir.

Dokunuşlardan gölgeler oluşur ve her çift kamera ile dokunmanın ve dokunmadaki basınç büyüklüğünün saptanabileceği bir üçgen oluştururlar. Bu şekilde dokunulan nokta belirlenir.

Bu teknolojinin, ölçeklenebilir olması, çok yönlü olması, özellikle daha büyük birimlerde uygun fiyatta olması nedeniyle kullanımı artmaktadır

4.4. ELT240320TP Dokunmatik Panelli LCD Ekran ve Özellikleri

Bu tezde kullanılan ELT240320TP Dokunmatik Panelli LCD Ekran ve özellikleri aşağıdadır. ELT240320TP Dokunmatik Panelli 2.8" TFT Renkli LCD 240 x 320 pikselden oluşan görüntüleme modülüdür. Bu ekranda ILI9320 grafik denetleyicisi ve kart üzerinde tek başına işleyebilen AD7843 dokunmatik panel denetleyicisi bulunmaktadır [5]. Şekil 4.3'te ELT240320TP modeli görülmektedir.

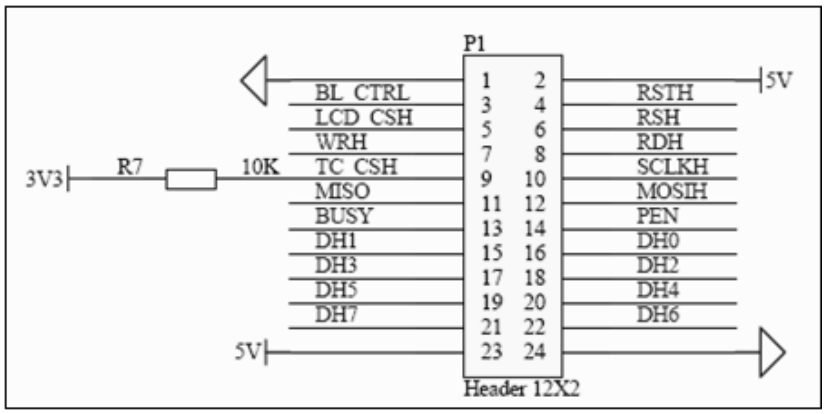


Şekil 4.3: 2.8" TFT Renkli LCD 240 x 320 piksel dokunmatik ekran

- 240x320 piksel dirençli-tür-2.8-inç dokunmatik ekran TFT LCD
- Gömülü LCD Sürücü (ILI9325) - 16-bit ya da 8-bit arayüz
- 4-telli Dokunmatik Ekran Denetleyici Sürücü (AD7843)
- 3.3V ya da 5V güç giriş ve arayüz logic [5].

4.4.1. ELT240320TP bağlantı şeması

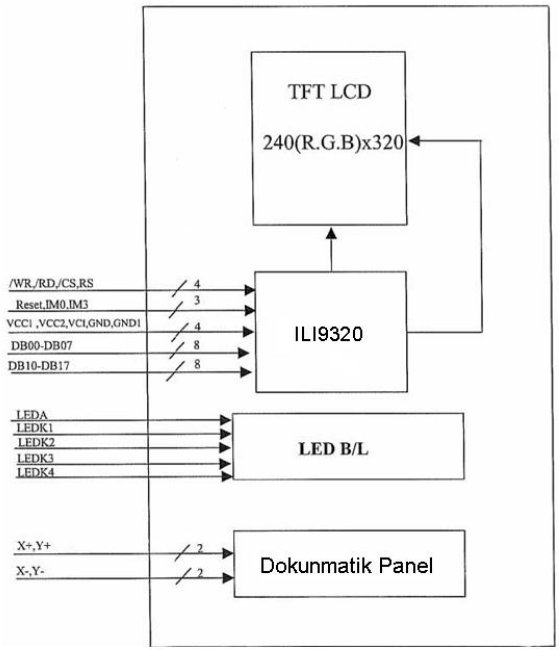
Şekil 4.4'de ELT240320TP dokunmatik panelli LCD ekranın yerleştirildiği plastik kartın uç giriş çıkışları gösterilmektedir.



Şekil 4.4: ELT240320TP Dokunmatik LCD Ekranın PCB kartındaki uç bağlantıları [25].

4.4.2. ELT240320TP blok biyagramı

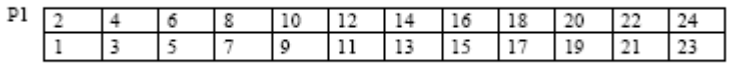
Şekil 4.5'te ELT240320TP dokunmatik panelli LCD ekranda bulunan ILI9320 grafik ekran sürücüsünü, arkaplan ışıklandırılmasını ve dokunmatik paneli içeren blok diyagram gösterilmektedir.



Şekil 4.5: ELT240320TP Blok Diyagramı [5].

4.4.3. ELT240320TP uç yerleşimi

Şekil 4.6'da ELT240320TP için kart üzerindeki uç yerleşimleri verilmiştir. LCD ekranın yerleştirildiği PCB Kartının üzerindeki P1 işaretinin anlamı uç1 olmadığına dikkat edilmelidir. Burada uç dizilimi şekil 4.6'da gösterildiği gibi sol alttan başlayarak sıralanmaktadır.



Şekil 4.6: ELT240320TP Dokunmatik LCD Ekranın Uç Yerleşimi

4.4.4. ELT240320TP uç giriş çıkışları

Tablo 4.1'de ELT240320TP için uç bağlantıları verilmiştir. Bu bağlantılardan TC_CS, SCLK, MISO, MOSI, BUSY, PEN dokunmatik panel girişleridir. 1,24 numaralı uçlar topraklama, 2,23 numaralı uçlar 5V gerilim ile beslenmelidir. Data0 - Data7 BL_CTRL, RSTH, LCD_CSH, RSH, WRH, RDH, uçlar ise LCD ekranın giriş çıkış uçlarıdır.

Dokunmatik Panel Giriş Çıkış Uçları

TC_CS : Dokunmatik Panel Chip Seçimi
SCLK : Dokunmatik Panel Saati Sinyali
MISO : Master In Slave Out Sinyali
MOSI : Master Out Slave In Sinyali
BUSY : Meşgul Bilgisi Sinyali
PEN : Çalışma Bilgisi Sinyali

LCD Ekran Giriş Çıkış Uçları

Data0 - Data7: Databus Bit 0-7: 8 Bit Paralel Komut-Veri Yolu Sinyalleri
BL_CTRL : Ekran Işığı Denetimi Sinyali

RSTH : Sıfırlama Sinyali

LCD_CSH : LCD Ekran Chip Seçimi Sinyali

RSH : Komut/Ekran Verisi Seçim Sinyali

WRH : Yazma Sinyali

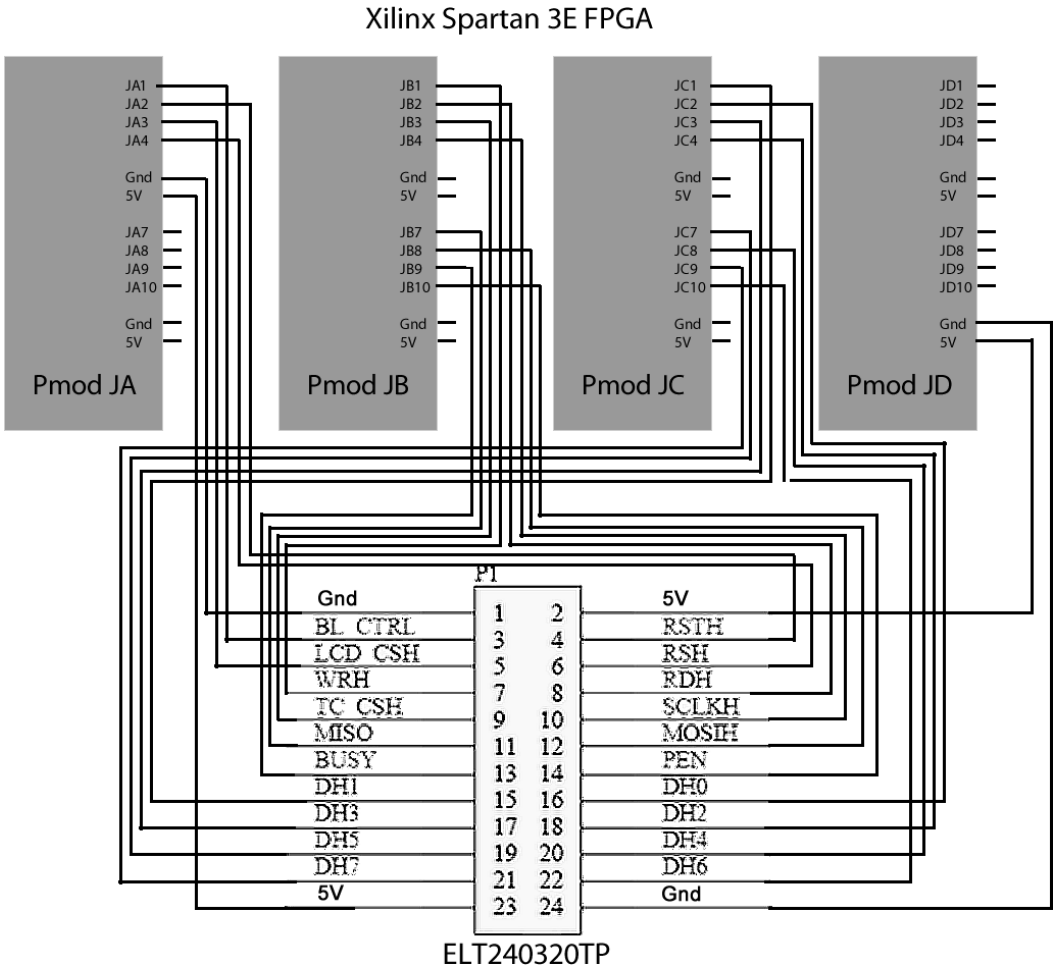
RDH : Okuma Sinyali

Tablo 4.1: ELT240320TP Dokunmatik LCD Ekranın Uç Bağlantıları [25].

PIN	NAME	Description	Command
1	GND		Gnd
2	5+		5+
3	BL_CTRL	Black Light	D5
4	RSTH	Reset	D2
5	LCD_CSH	CS	D1
6	RSH	RS	D0
7	WRH	Write	D3
8	RDH	Read	D4
9	TC CSH x		Touch Screen
10	SCLKH		Touch Screen
11	MISO		Touch Screen
12	MOSIH		Touch Screen
13	BUSY		Touch Screen
14	PEN		
15	DH1	Data 1	B1
16	DH0	Data 0	B0
17	DH3	Data 3	B3
18	DH2	Data 2	B2
19	DH5	Data 5	B5
20	DH4	Data 4	B4
21	DH7	Data 7	B7
22	DH6	Data 6	B6
23	5+		
24	GND		

4.5. FPGA Kartı ile LCD Ekran Bağlantıları

Nexys2 geliştirme kartı 6 uç iki satırdan oluşan 4 tane Pmod (çevresel bağlayıcı) sunmaktadır. Her bir pmod'da iki tane gerilim çıkışı, 2 tane topraklama girişi ve 8 veri sinyali bulunmaktadır. Tüm veri sinyallerinde kısa devre koruması bulunmaktadır ve Gerilim uçları atlama kablosu yardımıyla 3.3 volt ve 5 voltu desteklemektedir. Şekil 4.7'de FPGA kartının pmodları ile LCD ekranın PCB kartındaki uçları arasındaki bağlantılar gösterilmektedir.



Şekil 4.7: ELT240320TP Dokunmatik LCD Ekranın Uç Tanımlamaları

BÖLÜM 5. YAZILIM

Sistemin gerçekleştirilebilmesi için LCD Ekran IP çekirdeği ve Seri Haberleşme Birimi IP çekirdeği vhdl tabanlı olarak oluşturuldu. Bilgisayar ile veri iletişiminde MicroBlaze tabanlı c kodları kullanıldı. Bu bölümde öncelikle IP çekirdeği oluşturması ve kullanımı anlatılacak ardından hazırlanan IP çekirdekleri ve c tabanlı kodlar incelenecektir.

5.1. IP Çekirdeği Oluşturma

IP çekirdeği, çekirdekleri mantık tasarımında kullanılan, önceden test edilmiş çok karmaşık sistem seviyesindeki işlevlerdir. IP çekirdeği şu yararları sağlar: pazara daha hızlı sunum, geliştirme sürecinde sadelik ve kısalma, tasarım riski azalır, yazılım derleme işlemleri azaltır, doğrulama zamanı küçülür, tahmin edilebilir verim ve işlevsellik sağlar.

Donanımsal IP, önceden gerçeklenmiş mikroişlemci çekirdeği, gigabit alıcı verici, çarpıcı, toplayıcı, MAC birimi gibi öbekleri kapsar. Bu öbekler güç tüketimi, alan kullanımı ve başarım açılarından daha verimli olmayı sağlar.

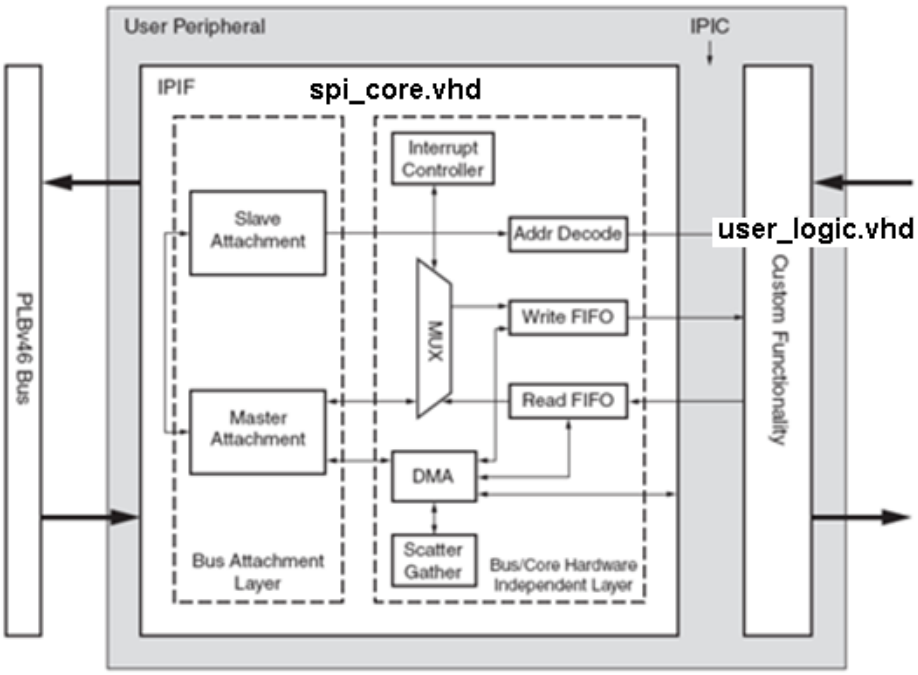
Yazılımsal IP, yüksek seviyeli işlevler için hazırlanmış kaynak kütüphanelerinin kullanıcı tarafından tasarıma eklenmesini tanımlar. Bu kütüphaneler PCI yolu arayüzü, DSP filtresi, PCMCIA arayüzü gibi karmaşık sistem seviyeli işlevler içerir [14].

Burada kendimize ait IP Çekirdeği oluşturma temelleri anlatılacaktır. IP Çekirdeği oluşturmak için Xilinx® Platform Studio (XPS) kullanılmıştır. XPS tasarım bölümlerinin çoğunu otomatikleştirmektedir bu nedenle XPS kullanarak bir gömülü işlemci oluşturmak basittir. Ve XPS'te bulunan Base System Builder (BSB) sihirbazı tasarım aşamasını birkaç adıma indirgemektedir [20].

BSB'yi çoğu gömülü işlemcinin tasarımında kullanabilmekteyiz. Temel işlemci tasarımı sonrasında XPS kullanarak sistemi özelleştiririz. Tasarımın özelleştirilmesi, var olan IP'ler üzerinde birkaç adımda yapılabilen değişikliklerle sağlanabilmektedir. Bunun yanında istenirse tamamen kendinize özgü karmaşık tasarımlar hazırlayarak oluşturulmuş tasarımlarla bütünleştirilebilmektedir.

IP oluşturmak için XPS'te bulunan CIP sihirbazı kullanılmaktadır. CIP sihirbazı iki defa hem IP oluşturmada hem de IP eklemede kullanılır. Birincisinde kendi çevresel birimimiz için kalıp oluşturacağız ve ikincisinde ise değiştirilmiş olan kalıbı sisteme ekleyeceğiz. Yeni bir çevresel birim eklenirken birim için bir isim, birimin hangi veri yoluna bağlanacağı ve hangi çevresel birim arayüzlerine hizmet edeceği belirlenmektedir. Bu seçeneklerden daha sonraki adımlarda ayrıntılı olarak bahsedilecektir. IP oluşturma işlemi sonunda CIP sihirbazı iki dosya oluşturur. Birincisi bizim belirlediğimiz arabirim ismine göre belirlenir ikincisi user_logic.vhd'dir. Biz buradaki dosya adlarını spi_core.vhd ve user_logic.vhd olarak kullanacağız. Birinci dosya spi_core.vhd hiyerarşik olarak user_logic.vhd ve user_logic.vhd'de tanımlanan modüllerin üstündedir.

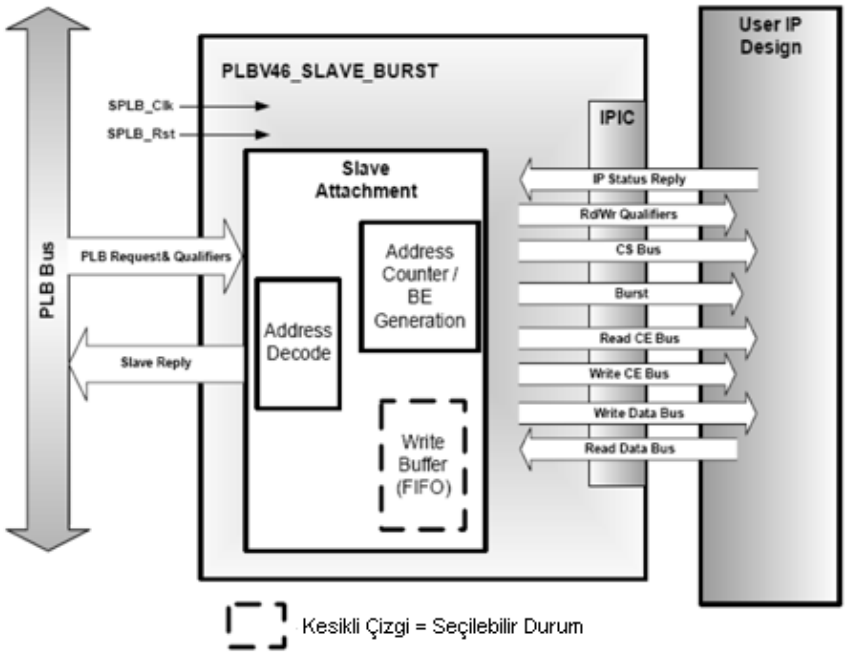
Şekil 5.1'de bulunan Custom Functionality bölümü user_logic.vhd'yi temsil etmektedir ve tüm bileşenler ise spi_core.vhd'ye eşittir [20].



Şekil 5.1: Veri yoluna bağlanmış bir çevresel birimin temel yapısı

Şekil 5.1'de mantık biriminin(user_logic.vhd) IPIF(Intellectual Property Interface-IP Çekirdeği Arayüzü) üzerinden OPB ya da PLB Bus'a bağlandığı görülmektedir. IPIC kısaltması IP InterConnect anlamına gelir ve kendi mantık birimimizin temsil ettiği sinyallere OPB ya da PLB Bus'tan daha kolay ulaşabilmesini sağlar. Şekil 5.1'deki diğer bölümlerde ise IPIF'nin desteklediği diğer özellikler bulunmaktadır.

Bu işlemler sonunda çevresel birimin kalıbı oluşturuldu. Bundan sonra bazı değişiklikler yaparak IP oluşturabilmektedir. Bunun için user_logic.vhd'ye mantık yapıları eklenmelidir. user_logic.vhd'ye kod yazılarak mantık yapısı eklenebilir yada istenirse yeni dosyalar eklenerek user_logic.vhd ile ilişkilendirilebilir. Bu durum Şekil 5.2'den anlaşılabilmektedir.



Şekil 5.2: Çevresel birimdeki kullanıcı mantığı hiyerarşisi

Tüm modüllerde gerçekleştirilmiş portlar harici sinyaller olmalıdır ve FPGA uçlarına uygun olarak UCF dosyasında tanımlanmalıdır.

Çevresel birim oluşturulduktan sonra CIP sihirbazı, ikinci kez içeri aktarma modunda çalıştırılmalıdır. Sonrasında sadece sihirbaza eklediğimiz dosyaları belirterek içeri aktarma işlemi yapılır. Bu şekilde yeni bir birim oluşturarak ve içeri aktarıldıktan sonra uygun veri yoluna bağlanarak işlem tamamlanır.

5.2. Seri Haberleşme Birimi IP Çekirdeği

Bu bölümde Serial Peripheral Interface(Seri Çevresel Arabirim) (SPI) Birimi'nin yapısı, çalışma şekli ve kullanımı anlatılacaktır. Burada oluşturulan seri haberleşme birimi Freescale firmasının üretmiş olduğu MC1322x adlı üçüncü-nesil ZigBee platformunda bulunan seri haberleşme biriminin özellikleri referans alınarak oluşturulmuştur [21].

SPI, sayısal(dijital) tüm devrelerin(bütünleşmiş devre-IC) seri haberleşmeleri için geliştirilmiş verimli haberleşme standartlarından biridir. Açılımı "Serial Perihpheral

Interface" olup Türkçesi " Seri Çevresel Arabirim"dir. SPI ismi Motorola tarafından bulunmuştur. SPI standardının bir diğer adı ise Microwire'dır ve bu Natiaonal Semiconductor firmasının tescilli ticari markasıdır.

SPI, çift yönlü olarak eş zamanlı çalışabilen (full duplex), verinin saat darbeleriyle birlikte eşzamanlı (senkron) olarak aktarıldığı bir seri haberleşme standardıdır. SPI Core seri belleklerle, çevresel birimlerle, ya da diğer işlem birimleriyle yüksek hızda eş zamanlı iletişim sağlayabilmektedir. SPI Core giriş ve çıkış yollarını eş zamanlı olarak kaydırılarak 1-32 bit arasında programlanabilen seri bit akışına imkân vermektedir. Bu haberleşmede veri(data, bitler) transferi master-slave ilişkisi ile gerçekleşir. Master veri haberleşmesini başlatan cihazdır. Master tarafından veri transferi başlatıldıktan sonra veri her iki yönde de eşzamanlı olarak aktarılabilir [22-24].

SPI ile ilişkili 4 uç bulunmaktadır. Bunlar SPI_SCK(clock-saat), SPI_MISO (MasterInSlaveOut-veri girişi), SPI_MOSI (MasterOutSlaveIn - veri çıkışı), SPI_SS(Slave Select-tümdevre seçimi). SPI_SCK hattında saat sinyalini Master birimi üretir ve bu sinyal tüm diğer slave cihazlara giriş olarak uygulanır. Master cihaz hangi slave'in SPI_SS ucunu aktif hale getirirse ilgili slave cihaz seçilmiş olur ve Master-Slave haberleşmesi gerçekleşir.

SPI saati çevresel birim saatinden referans alınarak oluşturulan bir saattir. (Kaynak olarak kullanılan FPGA saati 50MHz'dir) SPI'da bir ön düzenleyici sayesinde referans saat 2-256 arasında bölünerek 12.5 MHz ile 97.65 KHz arasında istenen saat değerine programlanabilmektedir.

MicroBlaze kullanıldığında CIP kullanımıyla yeni core eklenerek birden fazla birimle haberleşme sağlanabilir. Programlama söylemindeki kasıt 32 bitlik yazmaçlara SPI çalışma değerlerini girmektir.

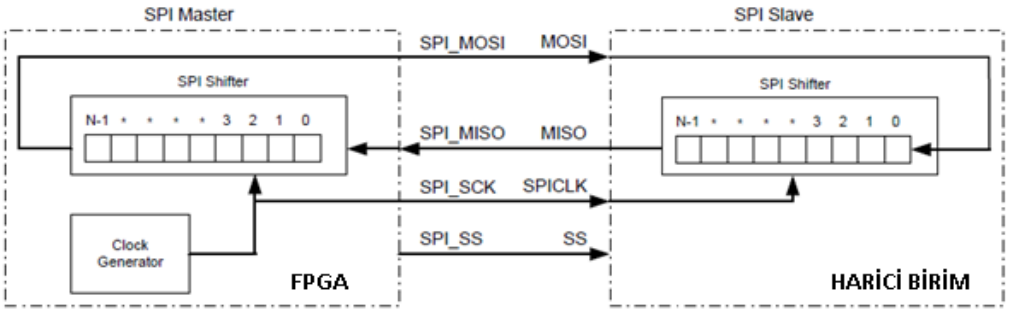
SPI_CORE IP Çekirdeğini oluşturmadaki amaç seri haberleşmeyi destekleyen tüm çevresel birimlerle haberleşmeyi sağlayan bir IP Çekirdeği sayesinde veri iletişimi sağlamaktır. SPI_CORE oluşturulduktan sonra kullanımı için sadece birkaç

değişkenin değerini atamak yeterli olacaktır. SPI_CORE kullanımı daha alt bölümlerde ayrıntılı olarak açıklanacaktır. SPI_CORE sayesinde her çevresel birimin kendine özgü zaman çizelgesine uygun olarak istenilen saat hızında, istenilen veri boyutunda, faz ve polarite seçimi yapılarak haberleşme sağlanabilir.

5.2.1. SPI birimi özellikleri:

• Master ya da slave çalışma modu
• 1 - 32 arasında programlanabilen transfer genişliği
• MSB-first shifting
• 4 byte (32 bit) genişliğinde veri tamponlama(buffer).(çift tamponlama yok)
• Programlanabilir bit iletim hızı (12 MHz en fazla)
• Seri saat fazı ve polaritesi seçenekleri
• Tam-Çift yönlü(çift yönlü olarak eş zamanlı çalışabilen) çalışma
• SPI iletişimi kesintisiz sürdürülebilir.
• Slave giriş/çıkış (input/output) seçimi
• SPI Slave etkinleşmesi için sistem saat gerekmemektedir.

5.2.2. SPI birimi sistem blok diyagramı



Şekil 5.3: SPI Sistemi Blok Diyagramı [21].

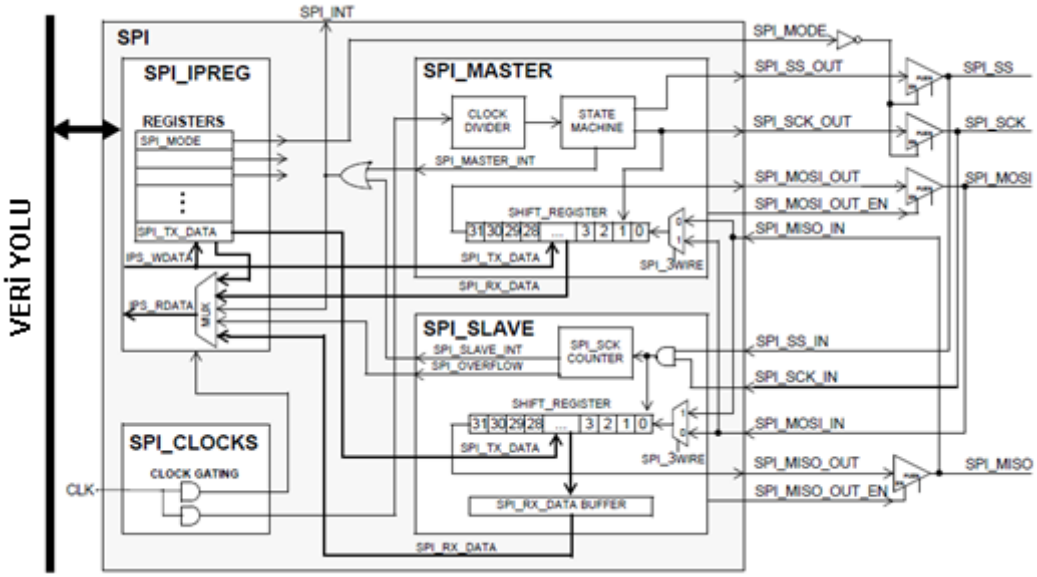
Şekil 5.3'te FPGA'yı SPI master olarak ve harici aygıtı iste SPI slave olarak göstermektedir. FPGA (master) tüm SPI veri iletimlerini başlatır. Bir veri iletimi sırasında master veri çıkışına (MOSI hattına) slave ise veri girişine (MISO hattına) eşzamanlı olarak veri kaydırır. SPI arabirimi master ile slave arasındaki eşzamanlı

veri değişimini desteklemektedir. SPI_SCK sinyali master için çıkış ve slave için giriş saatidir.

SPI birimi 32-bite kadar veri iletimini destekleyebilmektedir. SPI birimi sadece MSB-first olarak kaydırmayı desteklemektedir.

5.2.3. SPI birimi blok diyagramı

SPI biriminin üst düzey blok diyagramı Şekil 5.4' te gösterilmektedir. Burada master birimden saat eşliğinde eş zamanlı kaydırılarak slave birime bit transferi ve slave birimden dönen bitlerin yine eş zamanlı kaydırılarak yazmaçlara(registers) aktarılması gösterilmiştir



Şekil 5.4: Üst Düzey SPI Blok Diyagramı [21].

5.2.4. Fonksiyonel tanımlamalar

SPI kontrol, durum, veri tampon yazmaçlarını ve saat üretimini içerir. Master ve slave aynı saat üzerinden çalışmasına rağmen farklı işlemler yapabilirler. Devam eden bölümde SPI'ın fonksiyonları açıklanmaktadır.

5.2.4.1. Sinyal tanımlamaları

Tablo 5.1'de Seri Haberleşme IP çekirdeği için oluşturulmuş sinyaller ve açıklamaları gösterilmektedir.

Tablo 5.1: SPI Sinyal adlandırma ve tanımlamaları

Sinyal Adları	Yön	Etkin	Açıklama
SPI_SCK	Dijital Giriş/Çıkış	Programlanabilir	SPI port clock
SPI_MOSI	Dijital Giriş/Çıkış	High	SPI Port Master Çıkış Slave Giriş (MOSI) veri sinyali.
SPI_MISO	Dijital Giriş/Çıkış	High	SPI Port Master Giriş Slave Çıkış (MISO) veri sinyali.
SPI_SS	Dijital Giriş/Çıkış	Low	SPI Port Slave Seçimi (SS) sinyali.

5.2.4.2. Saat üretimi

Şekil 5.4'te SPI blok diyagramını saat bağlantısı gösterilmektedir. SPI birimi SPI_SCK'yı sadece master modunda üretir. SPI bit saat çevresel referans saatten oluşturulmaktadır(FPGA kaynaklı 50 Mhz). Çevresel referans saat ön ölçekleyici ile 1 ila 64 arasında bölünerek referans osilatör frekansı oluşturulmaktadır.

SPI biriminde çevresel referans saat sonradan ikinci bir ön ölçekleyici ile SPI_SETUP adlı SPI yazmacındaki SPI_SCK_FREQ[2:0] değerine göre programlanarak tekrar bölünmektedir.

3-bitlik SPI_SCK_FREQ[2:0] 0 ila 7 arasında programlanabilmektedir: SPI_SCK = (çevresel referans saat) / (2 ^ (SPI_SCK_FREQ + 1)) 24 Mhz'lik saat frekansını referans alarak SPI_SCK_FREQ[2:0] değerine göre programlanan SPI_SCK frekanslarının karşılıkları Tablo 5.2 'de gösterilmektedir.

Tablo 5.2: SPI_SCK_FREQ[2:0]'a göre SPI_SCK Frekans karşılığı

SPI_SCK_FREQ[2:0]	Divide Ratio	SPI_SCK_Frequency (referans frekans 25 Mhz)
0	2	12.5 mHZ
1	4	6.25 mHZ
2	8	3.12 mHZ
3	16	1.56 mHZ
4	32	780 kHZ
5	64	390 kHZ
6	128	195 kHZ
7	256	97.5 kHZ

Slave modunda, SPI_SCK, SPI master tarafından belirlenmektedir.

5.2.4.3. Temel işlem

SPI birimi master mod (SPI_MODE = 0) ve slave modda (SPI_MODE = 1) çalışabilecek şekilde tasarlandı. Her iki modda da veri aynı zamanda kaydırılır. SPI modülü, IP Bus (veri yolu) arabirimine yazılan 32 bit ile programlanır. Devamdaki başlıklarda SPI modülünün yapısı anlatılmaktadır.

5.2.4.4. SPI saatli kaydırma biçimleri

Farklı üreticilerden tarafından üretilen, geniş aralıklarda eşzamanlı seri çevresel birimlere uyum sağlamak için, SPI sistemi veri iletimi yapılırken dört saat formatından birinin seçiminde kullanılan bir saat (SPI_SCK) polarite biti, SPI_SCK_POL ve bir saat faz kontrol biti, SPI_SCK_PHASE bulunmaktadır.

SPI_SCK_POL seçime göre saat ile ters bir seri ekler. SPI_SCK_PHASE saat ve veri arasındaki iki farklı saat faz ilişkisinden birini seçiminde kullanılır

5.2.4.5. SPI kesmeler

SPI birimi SPI_STATUS yazmacındaki SPI_INT adlı tek kesme/durum bitine sahiptir. SPI_INT, SPI master modunda veri gönderildiğinde ve slave moda veri alındığında aktif (set) durumundadır. SPI_INT için herhangi bir yerel maskeleme bulunmamaktadır. Kesme işlemi Interrupt Controller (ITC) sayesinde etkin/devre dışı olarak kontrol edilebilmektedir.

5.2.5. SPI yazmaç bellek haritası

SPI IP Çekirdeği bellek eşlemeli yazmaçlar aracılığıyla programlanabilecek şekilde hazırlandı. Tablo 5.3'te yamaç adlandırmaları ve bit genişlikleri verilmektedir.

• Base adres - Eklenilen IP'ye sistem tarafından ayrılan adresin başlangıcıdır.
• Bu yazmaçlara okuma ve yazma için 32 bit olarak erişilebilir.

Tablo 5.3: SPI_CORE IP Çekirdeği Yazmaç Adresleri

Adresi	[31 to 0]	Erişim Türü	Genişliği
+0x00	SPI_TX_DATA (SPI yolu ile gönderilecek veri)	R/W	32
+0x04	SPI_RX_DATA (SPI yolu ile alınan veri)	R/W	32
+ 0x08	SPI_CLK_CTRL (SPI durumu ve saat kontrolü)	R/W	32
+ 0x0C (+ 12)	SPI_SETUP (SPI ayarları)	R/W	32
+ 0x10 (+ 16)	SPI_STATUS (SPI durumu)	R/W	32

5.2.6. SPI yazmaçları ve kontrol bitleri

5.2.6.1. Gönderilen veri yazmacı (SPI_TX_DATA)

Gönderilen veri yazmacı veri gönderiminde kullanılan 32-bitlik yazmaçtır. Master modda veri sola kaydırılarak (MSB-first) SPI_MOSI tarafından SPI_TX_DATA[31] ilk bit olarak gönderilir. Slave modda ise SPI_MISO tarafından SPI_TX_DATA[31] ilk bit olarak gönderilir.

- Yazmaç Adresi: Base+0x00
- Yazmaç Bit Genişliği: 0-31 bit

5.2.6.2. Alınan veri yazmacı (SPI_RX_DATA)

Alınan veri yazmacı (SPI_RX_DATA) veri alınırken kullanılan 32-bitlik yazmaçtır. Master modda veri sola kaydırılarak (MSB-first) SPI_MISO tarafından SPI_RX_DATA[0] ilk bit olarak gönderilir. Slave modda ise SPI_MOSI tarafından SPI_TX_DATA[0] ilk bit olarak gönderilir.

- Yazmaç Adresi: Base+0x04
- Yazmaç Bit Genişliği: 0-31 bit

5.2.6.3. Saat denetleme (SPI_CLK_CTRL)

Saat Control Register (Saat Denetleme Yazmacı) 8 bit SPI_SCK_COUNT, bir bit SPI_START ve 7 bit SPI_DATA_LENGTH değişkenlerinin atanmasında kullanılmaktadır.

SPI_SCK_COUNT [7:0] : Master modunda veri transferi yapılırken kaç tane SPI_SCK periyodu olacağını belirler. SPI_SCK periyotları SPI_SCK_COUNT+1'e eşittir ve transfer sırasında otomatik olarak indirilir. Slave modda etkin değildir.
Bit numarası: SPI_CLK_CTRL [8-15]

SPI_START: SPI_START'a 1 yazılarak SPI bekleme modundan çıkarak belirlenen saat eşliğinde veri transferine başlar.
Bit numarası: SPI_CLK_CTRL [2]

SPI_DATA_LENGTH[6:0] : SPI_DATA_LENGTH SPI_MOSI'ye kaç tane kaydırma yapılacağını belirlemede kullanılır. Kaydırılan veri sayısı SPI_DATA_LENGTH'e eşittir.
Bit numarası: SPI_CLK_CTRL [0-6]

- Yazmaç Adresi: Base+0x08
- Yazmaç Bit Genişliği: 0-31 bit

5.2.6.4. Kurulum yazmacı (SPI_SETUP)

SPI_MODE: SPI biriminin master ya da slave olarak seçiminde kullanılır. SPI_MODE low olduğunda SPI master modundadır. SPI_MODE high olduğunda SPI slave modundadır.
Bit numarası: SPI_SETUP[16]

SPI_SCK_FREQ [2:0] : Master modda SPI_SCK_FREQ değişkeni SPI_SCK frekansını kontrol eder. SPI_SCK = CLK / (2 ^ (SPI_SCK_FREQ + 1))
Bit numarası: SPI_SETUP[14 – 12]

SPI_MISO_PHASE : Master modda SPI_SCK'nın hangi kenarının SPI_MISO girişi sinyalini örnekleyeceğini belirler. Bu sayede veri aynı kenarda yada zıt kenarda tetiklenerek gönderilir.
Bit numarası: SPI_SETUP[10]

SPI_SCK_PHASE: SPI_MISO ve SPI_MOSI'ye bağlı olarak SPI_SCK fazını belirler
Bit numarası: SPI_SETUP[11]

SPI_SCK_POL: SPI_SCK polaritesini kontrol etmede kullanılır.
Bit numarası: SPI_SETUP[8]

SPI_SS_DELAY: SPI_SS'in başında ve sonunda +1 SPI_SCK periyodu kadar bekleme sağlar.
Bit numarası: : SPI_SETUP[3-2]

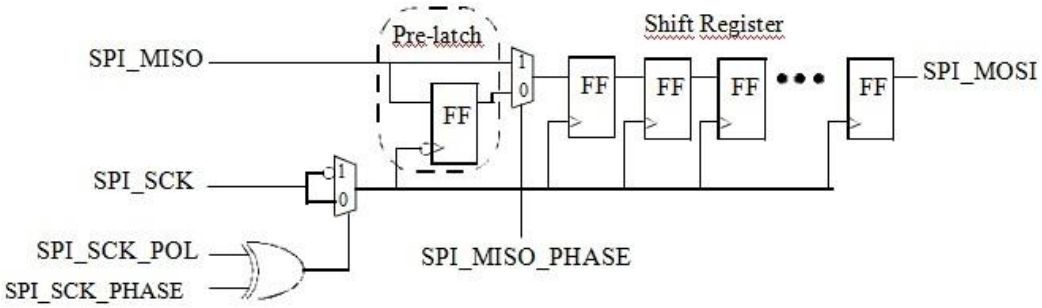
SPI_SS_SETUP: Master modda SPI_SS_OUT sinyalinin davranışını belirler. Eğer SPI_SS_SETUP otomatik olarak atanmışsa SPI_SCK nın ilk kenarına bağlı olarak etkinleşmektedir. SPI_SS_SETUP davranışları tablo 5.4'te gösterilmiştir.
Bit numarası: : SPI_SETUP[2-1]

- Yazmaç Adresi: Base+0x0C
- Yazmaç Bit Genişliği: 0-31 bit

Tablo 5.4: SPI_SS'ın SPI_SS_SETUP değerine göre davranış durumları [21].

SPI_SS_SETUP	SPI_SS_OUT (Master Modu)	SPI_SS_IN (Slave Modu)
00	Auto/Active High	Active High
01	Auto/Active Low	Active Low
10	Low	Active High
11	High	Active Low

Şekil 5.5'te Seri Haberleşme Biriminin MISO örneklemesi gösterilmektedir.



Şekil 5.5: SPI Master MISO örneklemesi [21].

5.2.6.5. Durum yazmacı (SPI_STATUS)

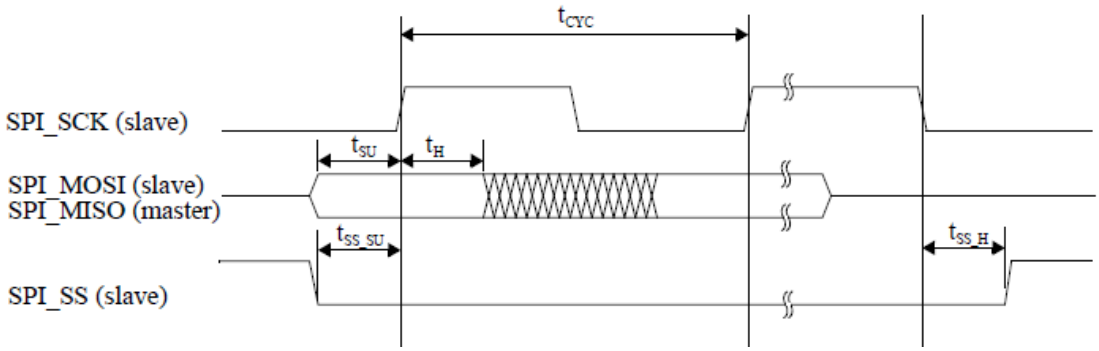
SPI_STATUS yazmacı veri transfer durum bilgisini vermektedir. SPI biriminin bulunduğu adım SPI_STATUS yazmacına yazılarak geri döndürülür.

- Yazmaç Adresi: Base+0x10
- Yazmaç Bit Genişliği: 0-31 bit

5.2.7. Zamanlama bilgileri

Şekil 5.6'da zamanlama bilgisi gösterilmektedir. SPI master saat FPGA kaynaklı 50 Mhz'den oluşmaktadır. Böylece en yüksek SPI_SCK frekansı 12,5 Mhz den oluşmaktadır. SPI slave ise gelen SPI_SCK ile saatlenmiştir. Zamanlama diyagramı sadece SPI I/O giriş(input) modundan olduğunda uygulanabilmektedir.

Başlama ve bekleme süreleri sadece diyagramdaki gösterildiği gibi yükselen kenarda değil aynı zamanda SPI_SCK'nın düşen ya da yükselen kenarlarına göre belirlenebilmektedir. Tablo 5.5'da SPI zamanlama sınırları gösterilmektedir.



Şekil 5.6: SPI IP Çekirdeği zamanlama diyagramı [21].

Tablo 5.5: SPI IP Çekirdeği zamanlama sınırları [21].

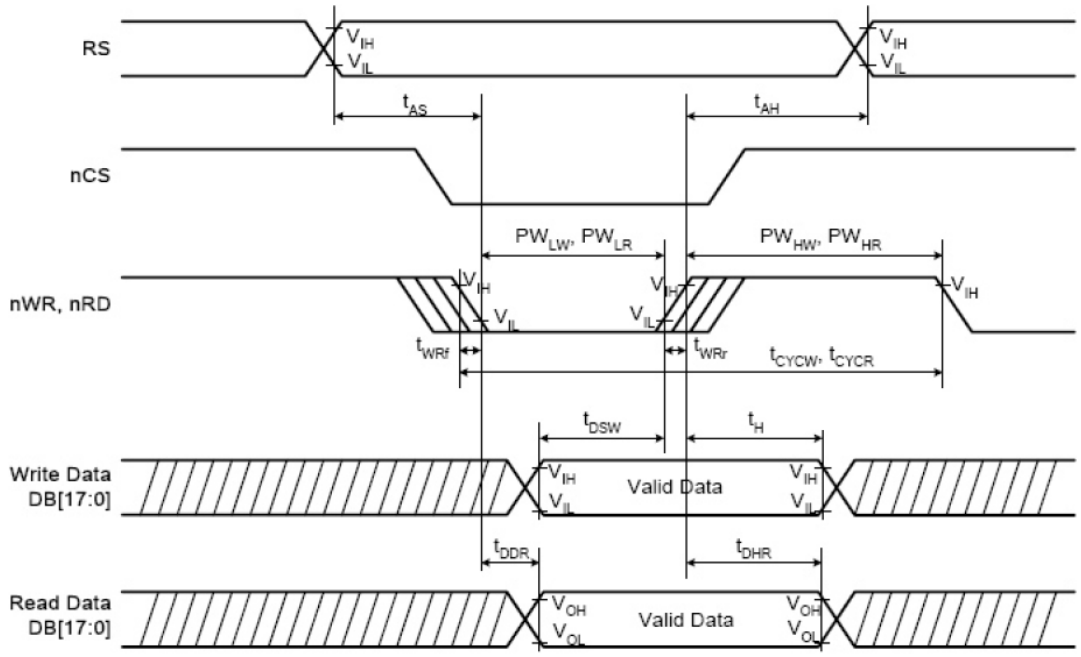
PARAMETRELER	GÖSTERİM	ÖZELLİKLERİ			BİRİM
		MIN	TYP	MAX	
Slave SPI_SCK Period	t_{CYC}		38		nsec
Slave SPI_MOSI & Master SPI_MISO1 Setup Time	t_{SU}		10		nsec
Slave SPI_MOSI & Master SPI_MISO1 Hold Time	t_H		10		nsec
Slave SPI_SS Setup Time	t_{SS_SU}		10		nsec
Slave SPI_SS Hold Time	t_{SS_H}		10		nsec

Oluşturulmuş olan SPI_ CORE IP Çekirdeğinin VHDL kodları tez sonunda Ek-A.2 de bulunmaktadır.

5.3. LCD Ekran IP Çekirdeği

LCD Ekran IP Çekirdeği (LCD_CORE) oluşturulurken kullandığımız ELT240320TP LCD ekranın 16 bit (RGB - Red(6) Green(5) Blue(5)) renk verisini 8 bit paralel LCD portuna iki aşamada gönderen VHDL kodu oluşturuldu. LCD_CORE ‘da Base+4 ya da Base+12 adresine 32 bitlik veri yazıldığında LCD_CORE etkinleşir. Ve veri yolunda (Bus2IP_Data) bulunan 32 bit veriyi alarak ilk 8 biti Bus2IP_Data(0 to 7) datah ‘a 8 ile 15 arasındaki 8 biti Bus2IP_Data(8 to 15) ise datal değişkenlerine alınır. Base+4’e yazılan değerler ekranda gösterilecek renk değerlerini içerir. Base+12 adresine yazılan değerler LCD ekran yönergeleridir(instuction). Değişkenlere alınan değerler yönerge veya renk bilgisi olarak ELT240320TP LCD ekranın yazma modu zaman çizelgesine göre adım adım gönderilir. Şekil 5.7’de LCD

Ekran Ip çekirdeğinin zaman çizelgesi, Tablo 5.6'da LCD Ekran IP çekirdeğini yazma modundaki zaman sınırlamaları gösterilmektedir.



Şekil 5.7: ELT240320TP Normal Yazma Modu Zaman Çizelgesi [5].

Tablo 5.6: ELT240320TP Normal Yazma Modu Zaman Sınırları [5].

Item		Symbol	Unit	Min.	Typ.	Max.	Test Condition
Bus cycle time	Write	t_{CYCW}	ns	100	-	-	-
	Read	t_{CYCR}	ns	300	-	-	-
Write low-level pulse width		PW_{LW}	ns	50	-	500	-
Write high-level pulse width		PW_{HW}	ns	50	-	-	-
Read low-level pulse width		PW_{LR}	ns	150	-	-	-
Read high-level pulse width		PW_{HR}	ns	150	-	-	
Write / Read rise / fall time		t_{WRr}/t_{WRf}	ns	-	-	25	
Setup time	Write (RS to nCS, E/nWR)	t_{AS}	ns	10	-	-	
	Read (RS to nCS, RW/nRD)			5	-	-	
Address hold time		t_{AH}	ns	5	-	-	
Write data set up time		t_{DSW}	ns	10	-	-	
Write data hold time		t_H	ns	15	-	-	
Read data delay time		t_{DDR}	ns	-	-	100	
Read data hold time		t_{DHR}	ns	5	-	-	

LCD ekrana veri gönderme işlemi yaklaşık 200 ns içinde tamamlanır. Ancak oluşturduğumuz sistemde her veri gönderme işlemini microblaze yönlendirdiği için ekrana 16 bit veri gönderimi microblaze'in hızına göre yani 1 us 'ye yakın sürede gerçekleşir. ELT240320TP 320x240 piksele sahiptir. Biz LCD_CORE sayesinde

ekranda 65536 farklı renkten oluşan saniyede 17 resim gösterebilmektedir. Bu hız bizim gerçekleştirdiğimiz insan makine arabirimi için yeterli olmaktadır.

Özetle LCD_CORE kendisine gelen veriyi high ve low olarak ikiye ayırır ve LCD ekranın zamanlamasına uygun olarak ekrana bu verileri paralel 8 bit olarak gönderir.

Oluşturulmuş olan TC_LCD_CORE IP Çekirdeğinin VHDL kodları tez sonunda Ek-A.1 de bulunmaktadır.

5.4. MicroBlaze Tabanlı C Kodları

Bu bölümde XPS kullanılarak microblaze tabanlı tasarlanan sistemin yapısı anlatılacak ve kullanılan yazılımlardan bahsedilecektir. Microblaze tabanlı sistem tasarlanılırken önce BSB kullanılarak yeni bir sistem oluşturuldu. Kart seçim sayfasında Digilent Nexys2-500 geliştirme kartı seçildi. Tek işlemcili (MicroBlaze) bir sistem 50 Mhz. Saat hızında 16 KB yerel hafıza ayrılarak seçildi. Çevresel birim ayarları sayfasında 16 MB Micron_Ram ile saniyedeki bit sayısı 115200 veri bitleri 8 olan RS232 bağlantısı eklendi. BSB sayesinde sistem için gerekli olan temel dosyalar oluşturuldu ve eklenen tüm birimlerin adreslemeleri otomatik olarak atandı.

BSB ile sistem oluşturulduktan sonra CIP sihirbazı kullanarak oluşturmuş olduğumuz SPI_CORE ve TC_LCD_CORE IP Çekirdeği IP kataloğuna getirilerek mb_plb veri hattına eklendi adres atamaları gerçekleştirildi. Bu sayede oluşturduğumuz IP Çekirdeği MicroBlaze işlemcisinin kontrolünde kullanılabilir hale getirildi. Oluşturulan IP çekirdeklerine hazırlanan c tabanlı kodlar ile erişim sağlanabilmektedir. Aşağıda sistemde kullanılan c tabanlı yazılımsal kaynaklar anlatılmaktadır. Yazılım aşamasında oluşturulan c kodları.

5.4.1. Ana program (Main.c)

Ana program olan main.c adlı kaynak dosyada RS232'den gelen verileri parçalarına ayırarak işlenilebilir fonksiyonlar haline getirilir. Burada seri porttan alınan verilen bir değişkende depolanır. Depolanan veriden öncelikli olarak fonksiyon adı alınır.

Sonrasında sırasıyla parametreler gelen parametre adına göre adlandırılmış değişkenlere aktarılır. Alınmış olan fonksiyon parametrelerine uygun olarak LCD ekranda çalıştırılır. "Main.c" dosyasındaki kodlar Ek-B.1'te verilmiştir.

5.4.2. Dokunmatik panel fonksiyonları (TC240320TP.c)

TC240320TP.c adlı kaynak dosya dokunmatik panel fonksiyonlarının tanımlandığı dosyadır. Dokunmatik panel fonksiyonları, seri olarak haberleşen AD7843 analog dijital dönüştürücüden aldıkları X,Y konum verilerini döndürürler. Burada bulunan getTouchControlX () fonksiyonu x konumundaki dokunmayı getTouchControlY () fonksiyonu y konumundaki dokunmayı algılamamızı sağlamaktadır.

TC240320TP.c dosyasında bulunan kodlar EkB-2'te verilmiştir

5.4.3. LCD Ekran fonksiyonları (LCD240320TP.c)

lcd240320tp.c adlı kaynak dosya grafik LCD fonksiyonlarının tanımlandığı dosyadır. Bu dosyadaki fonksiyonlar ThaiEasyElec firmasının Pic denetleyicileri için oluşturmuş olduğu fonksiyonlar [25] temel alınarak oluşturulmuştur. LCD fonksiyonları:

- SetBackGroundColor: Ekranı tek renk (RGB) ile doldurur.
- SetFontColor: Yazı rengini belirler.
- SetBackColor: Yazı arkaplan rengini belirler.
- TSLCDSetMargins: Verilen boyutlara göre ekranın boyutları belirlenir.
- TSLCDSetMarginsDefault: Varsayılar ekran boyutları uygulanır.
- CharDisp: Verilen karakteri ekranda istenilen satır ve sütuna yazdırır.
- TSLCDOutDat: Görüntü verisi gönderimini sağlar
- TSLCDOutIns: Komut bilgisi gönderimini sağlar
- TSLCDRst: Ekranın sıfırlanmasını sağlar
- TSLCDInit: Ekranın ön işlemlerini gerçekleştirerek ekranı hazır hale geririr.

- FillRect: X,Y koordinatlarına gore konumu verilen dikdörtgeni istenilen renkte(RGB) çizer.
- ShowPic: X,Y koordinatlarına gore konumu verilen bölüme istenilen renkleri(RGB) RAM ‘ deki adreslerden alarak gösterir.
- PrintStr: Verilen String ifadeyi ekranda istenilen satır ve sütuna yazdırır
- FillCirc: X,Y koordinatlarına gore konumu ve yarıçapı (rad) verilen verilen bir çemberi çizer

LCD240320TP.c dosyasında bulunan kodlar EK-B.3’te verilmiştir

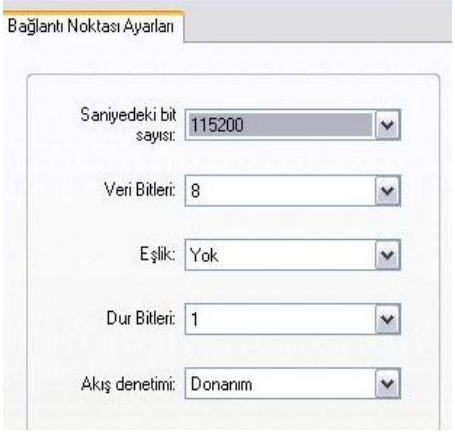
BÖLÜM 6. UYGULAMA

Bu tezde oluşturulan sistem sayesinde gömülü sistem tasarımı, dokunmatik ekranlı HMI tasarımının da kullanılan temel fonksiyonların Terminal ve Adobe Air yazılımı aracılığıyla LCD Ekrana göndererek gerçekleştirilir. Burada sistem LCD ekran, FPGA ve PC olarak üç bölümden oluşmaktadır. Tasarımcının yazmış olduğu temel grafik fonksiyonları FPGA tarafından işlenerek LCD ekranda çalıştırılır. Tasarımcı bu sayede yazılımı derleme ve FPGA'ya yükleme aşamalarından kurtulmuş olur. Terminal bağlantısı ve sistemde kullanılabilen örnek grafik fonksiyonları Adobe Air ile tasarım örnekleri sonraki başlıklar altında açıklanmıştır.

6.1. Hiper Terminal Bağlantı Ayarları

Bilgisayar ile FPGA kartının haberleşmesi için seri port bağlantıları yapıldıktan sonra aşağıdaki terminal ayarları yapılmalıdır. Bu ayarlar örnek ekran fonksiyonlarının çalıştırılabilmesi için gerekmektedir. Hiper terminal ayarları için:

Windows Windows NT dizininde bulunan hiper terminal "Windows NT\hypertrm.exe" adlı exe dosyası ile çalıştırılmaktadır. Bağlantı olarak com1 seçilmelidir, bir sonraki Bağlantı Noktası Ayarları sayfasında (Şekil 6.1'deki sayfa):



Şekil 6.1: Terminal Bağlantı Noktası Ayarları

Saniyedeki bit sayısı: 115200,
Veri bitleri: 8 bit,
Eşlik: yok,
Dur bitleri: 1,
Akış Denetimi. Donanım olarak seçilir ve ayarlar uygulanmalıdır.
Terminal sayfasında yazılan karakterleri yerel yankılandır (Terminal/ Dosya/ özellikler/ ayarlar/ ascii ayarları) seçeneğini etkinleştirilmelidir

Bu ayarlar yapıldıktan sonra terminal sayfası açılamalıdır. FPGA kartı ve seri port bağlantısı doğru yapılmış ise sistem FPGA kartına yüklendiğinde Terminal Sayfasında "FPGA' ya Sistem başarıyla yüklendi. Bir Fonksiyon Giriniz ve 'Enter' Tuşuna Basınız. " mesajları görüntülenecektir. Bu mesajdan sonra girilen fonksiyonlar işletilir ve LCD ekranda sonuçları görüntülenebilir. Her fonksiyon işletildikten sonra "Fonksiyon Başarıyla Çalıştırıldı" mesajı Terminal Sayfasında belirtilmektedir. Bu aşamadan sonra sistemde LCD Ekran için hazırlanmış olan ilkel grafik fonksiyonlarından istenilen fonksiyonlar çalıştırılabilir.

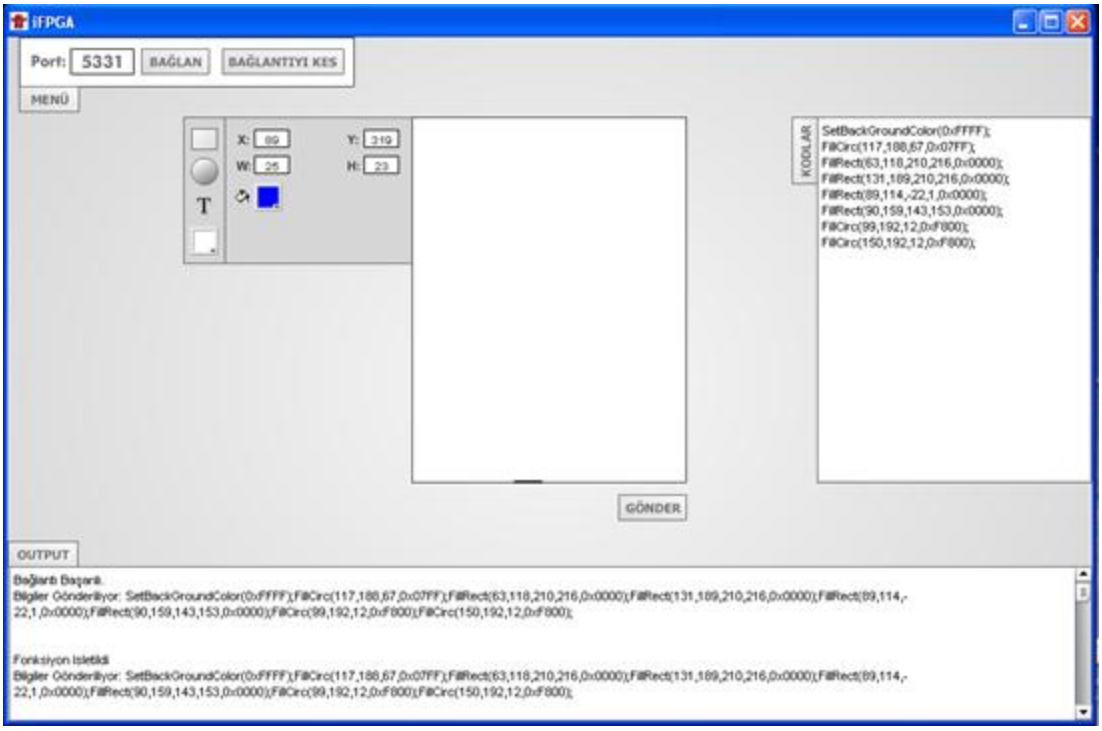
6.2. Ekran Tasarımında Kullanılan Yazılım

Flash platform çalıştırıcılarından biri olan Adobe AIR®(Adobe Integred Runtime), geliştiricilerin kapsamlı içerikleri, uygulamaları ve videoları olabilecek en geniş hedef kitlesine tutarlı bir biçimde ulaştırmasına olanak verir [26].

Platform bağımsız olarak çalışabilen Adobe AIR ile web ActionScript, JavaScript, HTML kullanarak masaüstü uygulamalar geliştirilebilmektedir [27].

Biz burada AIR platformundan yararlanarak oluşturduğumuz yazılım sayesinde ekran tasarımı yapabilmeyi ve yapılan tasarımın fonksiyonlara dönüştürülerek bir ek yazılım aracılığı seri porta gönderilmesini sağladık. Bu sayede tasarımcı doğrudan tasarlanılan birimin üzerine yoğunlaşabilmektedir.

Bilgisayar ortamında tasarım yapabilmek için oluşturduğumuz yazılım Şekil 6.2'de gösterilmektedir. Oluşturulan yazılım bölümleri daha sonraki sayfalarda açıklanacaktır.



Şekil 6.2: Ekran tasarımında kullanılan yazılım

Oluşturulan yazılımda öncelikle port numarası girilerek seri porta bağlanılmaktadır. Bu yazılımda istenilen şekil araç kutusundan seçilerek LCD Ekran için ayrılmış tasarım bölümünde çizilebilmektedir. Burada çizilen her şekil sağda bulunan kodlar listesine eklenmektedir ve eklenen herhangi bir şekil yer değiştirdiğinde kod listesidede güncellenmektedir. Çıktı ekranında ise bağlantı durumu ve gönderilen fonksiyonları işletilme bilgisi görüntülenmektedir.

Ekran Tasarımı tamamlandığında son aşama olarak gönder tuşu ile oluşturulan LCD Ekran fonksiyonları LCD Ekranda görüntülenmektedir. Bilgisayardaki yazılım ile tasarlanan ekran gerçek cihaz üzerinde test edilebilmektedir.

6.3. LCD Ekran Fonksiyonlarının Çalışma Şekli

Aşağıdaki fonksiyonlar LCD ekranda arayüz tasarımı için kullanılan grafik fonksiyonlarıdır. Bu fonksiyonlar RS232 yoluyla terminal bağlantısından gönderilir. Gelen fonksiyon microblaze üzerinde yazılan c kodları ile bileşenlerine ayrılarak LCD ekrana gönderilir. FPGA kartına bağlı olan Dokunmatik LCD ekranda işlenmiş fonksiyonlar gösterilir. Bu tezde test için oluşturulan fonksiyonlar ilkel grafik fonksiyonlarından bazılarıdır. Aşağıda bu fonksiyonların açıklamaları ve kullanımından bahsedilecektir.

Bu fonksiyonları çalıştırabilmek için öncelikle FPGA kartı – TCLCD ekran bağlantıları yapılır. Karttaki FPGA'ya oluşturulmuş olan sistem yüklenir. Bilgisayara FPGA kartı seri port aracılığıyla fiziksel olarak bağlanır. Terminal ayarları Terminal bölümünde anlatılan ayarlara göre yapılır ve oluşturulan Terminal Bağlantısı açılır. Açılan sayfada aşağıdaki fonksiyonlar uygun değerler verilerek Enter tuşu ile işletilebilir.

6.4. LCD Örnek Ekran Fonksiyonları ve Tanımlamaları

Bu bölümde sistemin çalışırlığını test edebilmek için bazı ilkel grafik fonksiyonları oluşturuldu. Bu fonksiyonlar ve işlevleri aşağıda anlatılmaktadır.

a) Arkaplan Renk Seçimi: Ekranı tek renk(RGB) ile doldurur.
   SetBackGroundColor(word color);

b) Dikdörtgen Çizimi: sx, sy konumundan ex, ey konumuna kadar istenilen renkte(RGB) dikdörtgen çizer.
   FillRect(ts_pos_t sx, ts_pos_t ex, ts_pos_t sy,ts_pos_t ey,word color);

c) Resim Ekleme: sx, sy konumundan ex, ey konumuna kadar istenilen renkleri(RGB) RAM ' deki adreslerden alarak gösterir.
   ShowPic(ts_pos_t sx, ts_pos_t ex, ts_pos_t sy, ts_pos_t ey, const unsigned short *pic);

d) Yazı Rengi Belirleme Yazı rengini belirler.
SetFontColor(word color);

e) Yazı Arkaplan Rengi Belirleme Yazı arkaplan rengini belirler.
SetBackColor(word color);

f) String Yazdırma Verilen String ifadeyi ekranda istenilen satır ve sütuna yazdırır
PrintStr(byte line, byte column, char *str);

g) Karakter Yazdırma Verilen karakteri ekranda istenilen satır ve sütuna yazdırır.
CharDisp(char character, ts_pos_t sx, ts_pos_t sy);

h) Çember Çizimi cx konumu ve cy konumu ve yarıçapı (rad) verilen bir çemberi çizer
FillCirc(ts_pos_t cx, ts_pos_t cy, ts_pos_t rad, word color);

6.5. Örnek Uygulamalar

6.5.1. Uygulama 1

Burada arabirim tasarımı için oluşturulmuş ilk örnek uygulama verilmiştir. Bu uygulamada ekran tasarımı terminalden gönderilem komutlarla yapılmıştır. Şekil 6.3'te kullanılan ekran fonksiyonları görüntülenmektedir.

```
//arkaplan
ShowPicRAM(0,240,0,320,0x00000000);


//sarj gostergesi
FillRect(220,228,290,291,0xFFFF);
FillRect(220,228,294,295,0xFFFF);
FillRect(220,228,298,299,0xFFFF);
FillRect(220,228,302,303,0xFFFF);
FillRect(220,228,306,307,0xFFFF);

FillRect(222,226,280,287,0xFFFF);
FillRect(223,225,281,286,0x0000);

//cekim gostergesi
FillRect(12,20,290,291,0xFFFF);
FillRect(12,20,294,295,0xFFFF);
FillRect(12,20,298,299,0xFFFF);
FillRect(12,20,302,303,0xFFFF);
FillRect(12,20,306,307,0xFFFF);

FillRect(14,18,280,287,0xFFFF);
FillRect(15,17,281,286,0x0000);
```

Şekil 6.3: Terminal ekranına girilen fonksiyonlar

Şekil 6.4'te ise terminale gönderilmiş olan örnek fonksiyonların LCD Ekranda çalışan hali görüntülenmektedir.



Şekil 6.4: Fonksiyonların LCD ekranda işlenmiş hali

6.5.2. Uygulama 2

Burada arabirim tasarımı için oluşturulmuş ikinci örnek uygulama verilmiştir. Bu uygulamada örnek bir menü tasarlanmıştır. Şekil 6.5'te arabirim tasarlanırken kullanılan örnek ekran fonksiyonları görüntülenmektedir.

```
SetBackGroundColor(0xDE00);
SetBackColor(0xDE00);
SetFontColor(0x0000);
PrintStr(18,3,"MENU");
//ust cizgi
FillRect(25,215,280,285,0x0000);

//1. satir
FillRect(25,75,220,260,0x0000);
FillRect(30,80,225,265,0xF800);
FillRect(85,135,220,260,0x0000);
FillRect(90,140,225,265,0xF800);
FillRect(145,195,220,260,0x0000);
FillRect(150,200,225,265,0xF800);

//ikinci satir
FillRect(25,75,155,195,0x0000);
FillRect(30,80,160,200,0xF800);
FillRect(85,135,155,195,0x0000);
FillRect(90,140,160,200,0xF800);
FillRect(145,195,155,195,0x0000);
FillRect(150,200,160,200,0xF800);

//3. satir
FillRect(25,75,95,135,0x0000);
FillRect(30,80,100,140,0xF800);
FillRect(85,135,95,135,0x0000);
FillRect(90,140,100,140,0xF800);
FillRect(145,195,95,135,0x0000);
FillRect(150,200,100,140,0xF800);

SetBackColor(0xF800);
SetFontColor(0xFFFF);
PrintStr(15,3,"Mesajlar");
PrintStr(11,3,"Medya");
PrintStr(7,3,"Yukleme");
PrintStr(15,8,"Rehber");
PrintStr(11,8,"Ajanda");
PrintStr(7,8,"Indirme");


PrintStr(15,14,"Islem K.");
PrintStr(11,14,"Ofis");
PrintStr(7,14,"Yardim");

//alt cizgi
FillRect(25,215,45,50,0x0000);

SetBackColor(0xDE00);
SetFontColor(0x0000);
PrintStr(1,14,"CIKIS");
```

Şekil 6.5: Terminal ekranına girilen fonksiyonlar

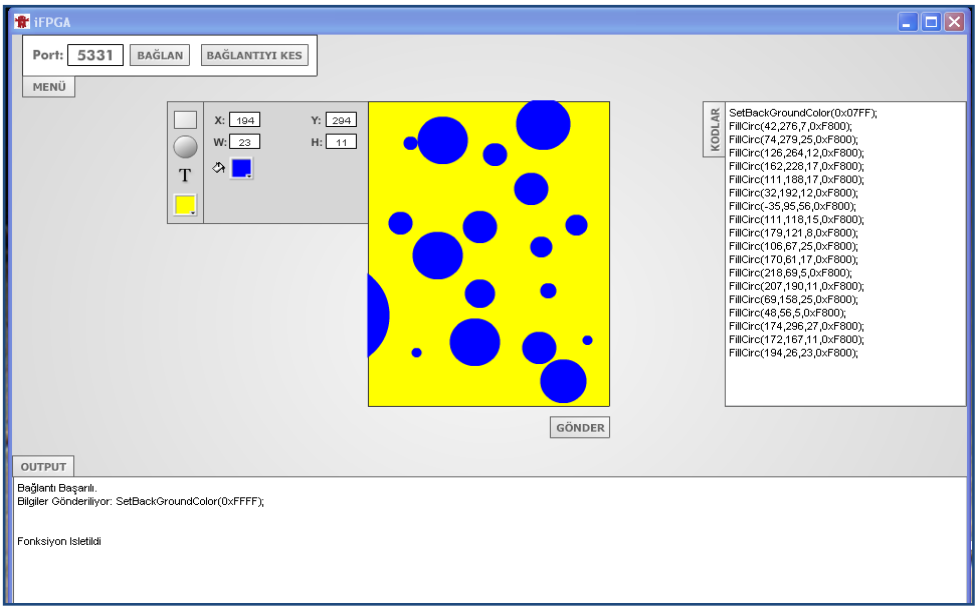
Şekil 6.6'da ise terminale gönderilmiş olan ikinci örnek fonksiyonların LCD Ekranda çalışan hali görüntülenmektedir.



Şekil 6.6: Fonksiyon LCD ekranda işlenmiş hali

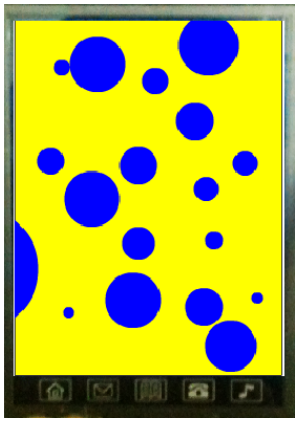
6.5.3. Uygulama 3

Air platformunda yazılan yazılım sayesinde tasarım görsel olarak tasarlanabilmektedir. Şekil 6.7'de tasarlanan ilk örnek şekil gösterilmektedir. Yazılım kullanılırken önce seri port bağlantısı yapılır. Tasarım gerçekleştirildikten soran "gönder" tuşuna basıldığında oluşturulan tasarım LCD Ekranda görüntülenebilmektedir.



Şekil 6.7: Ekran tasarımının yapıldığı yazılım

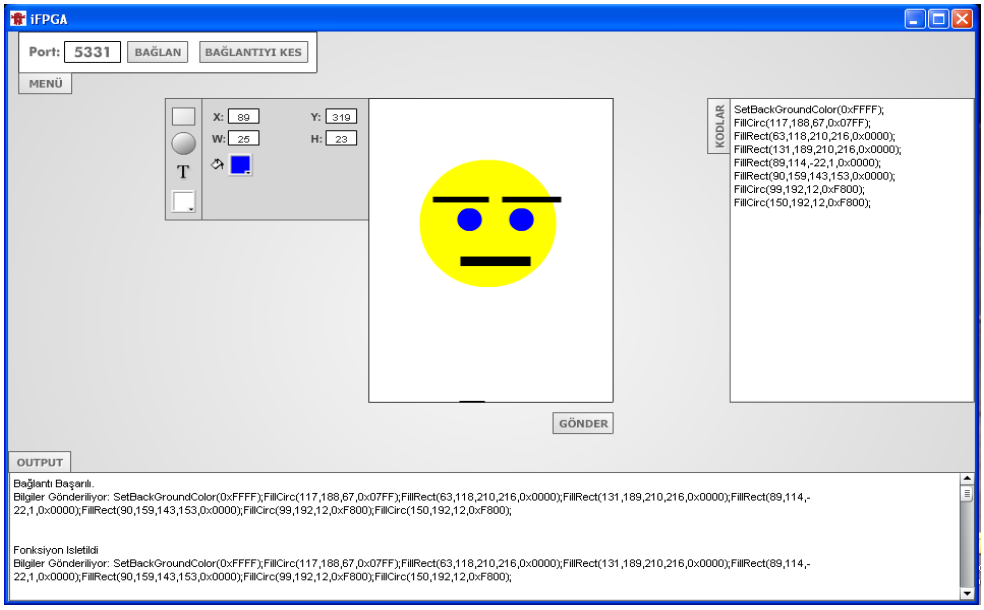
Şekil 6.8'de ise yazılımdan gönderilmiş olan ilk örnek fonksiyonların LCD Ekranda çalışan hali görüntülenmektedir.



Şekil 6.8: Fonksiyon LCD ekranda işlenmiş hali

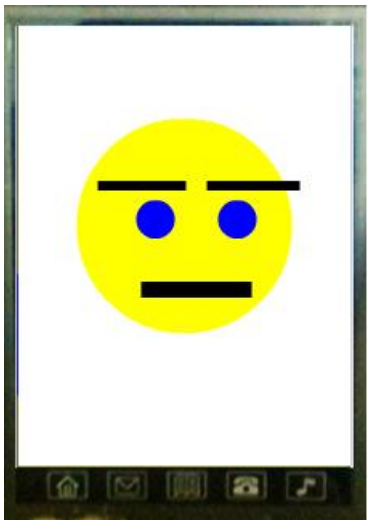
6.5.4. Uygulama 4

Air platformunda yazılan yazılım sayesinde bilgisayardan tasarlanan şekil LCD ekrana gönderilmiştir. Şekil 6.9'da tasarlanan örnek ikinci şekil gösterilmektedir.



Şekil 6.9: Ekran tasarımnın yapıldığı yazılım

Şekil 6.10'da ise yazılımdan gönderilmiş olan ikinci örnek fonksiyonların LCD Ekranda çalışan hali görüntülenmektedir.



Şekil 6.10: Fonksiyon LCD ekranda işlenmiş hali

Uygulamalardanda anlaşıldığı gibi hiper terminale girilen kodlar Enter tuşuna basıldığında LCD ekranda çalıştırılmaktadır. Bu işlemler sayesinde tasarımcılar gerçek cihazda ekran tasarımı, yani fonksiyonların doğrudan tasarlanılan ekranda çalışmasıyla arabirim tasarımını yapabilmektedirler.

BÖLÜM 7. SONUÇLAR VE ÖNERİLER

7.1. Sonuçlar ve Öneriler

Bu tezin amacı olan insan makine arabirimi tasarım aşamasını bilgisayar ortamına aktarma ve bu tasarımın gerçek cihazlarla yapılması, oluşturulan sistem sayesinde sağlanmıştır. Bu sistemde örnek olarak oluşturulan ekran fonksiyonları test edilmiş ve doğrulanmıştır. Oluşturulan sistemin arabirim tasarımını oldukça kolaylaştırdığı görülmüştür. Bunun yanında sentezleme ve yükleme adımlarının olmayışı arabirim tasarım sürecisi önemli ölçüde hızlandırmıştır. Oluşturulan bu sistemde:

İlk olarak hedeflenen arabirimin görsel arayüzünü hızlı olarak tasarlayabilme adımı olan terminal yoluyla girilen fonksiyonların doğrudan FPGA'da işlenmesi ve LCD ekran da görüntülenmesi doğrulanmıştır.

İkinci olarak hedeflenen arabirimi yeni bir sistemle kolayca bütünleştirebilme için hazırlanmış olan IP çekirdekleri sayesinde farklı FPGA'larda sistem kolayca bütünleştirebilmesine imkân tanıdığı yeni bir gömülü sistem oluşturularak doğrulanmıştır.

Bu tezde bilgisayar ortamında arabirim tasarımının temeli atılmıştır. Bu temel üzerine Adobe Air geliştirme ortamı ile hazırladığmız arayüz tasarım yazılımı sayesinde seri porta ekran fonksiyonları gönderilerek istenilen arabirim tasarımı yapılabilmektedir. Yani sistem tasarımcısı bilgisayarda kullandığı bir yazılım sayesinde görsel araçlarla insan makine arabirimini kolayca tasarlayabilir hale gelmiştir.

Bu çalışma sayesinde FPGA tabanlı olarak hazırlanan gömülü sistem arabirimlerinin bilgisayar ortamında tasarlanması arabirim tasarımını kolaylaştırmış ve tasarıma harcanan zamandan kazanç sağlanmıştır. Bunun yanında görsel bir arayüz sayesinde arabirim tasarımının daha da kolaylık sağladığı bu çalışmadan anlaşılmıştır.

Sonraki çalışmalarda kullanıcı arabirim tasarlanması işleminin tümünü, bilgisayar üzerindeki görsel araçlar kullanarak, gerçek cihaz üzerinde tasarlayabilmeyi sağlayan sistemi oluşturma hedeflenmektedir.

KAYNAKLAR

[1]Niladri, Roy, Aralık 2009, *Embedded Display Control Applications Using FPGAs* [çevrimiçi], Lattice Semiconductor Corporation, http://www.techfocusmedia.net/ fpgajournal/feature_articles/20091201-lattice/ **(Ziyaret tarihi: 12 Eylül 2010)**.

[2] Williamson, J., Li-Brouwer A., "Implementing an FPGA-based Human-Machine Interface For Home Appiliances", ***Altera Corporation****,1-6,* 2009

[3] Altera, "Implementing a Cost-Effective Human-Machine Interface for Home Appliances" ***Altera*** **Corporation*, WP-01083-2.0,*** 1-8, (2009)

[4] Altera,"Using FPGAs to Render Graphics and Drive LCD Interfaces", ***Altera*** **Corporation*, WP-01100-1.0,*** 1-4, (2009)

[5] Thaieasyelec, Ekim 2007**,** 2.8" TFT LCD withTouch Screen [çevirimiçi], http://www.optimal-microsystems.com/store/ThaiEasyElec/pdf/ELT240320TP.pdf **(Ziyaret tarihi: 22 Mayıs 2010)**

[6] Digilent Inc, 2010, Nexys2 FPGA Board, digilentinc.com, http://www.digilentinc.com/Products/Detail.cfm?Prod=NEXYS2 **(Ziyaret tarihi: 10 Nisan 2010)**

[7] Xilinx, Inc., "*MicroBlaze Processor Reference Guide* EDK 10.1i", ***Xilinx, UG081 (v9.0)***,17-28,(2009).

[8] Xilinx, Inc., "Embedded System Tools Reference Guide EDK 11.3.1", ***Xilinx, UG111***,17-28,(2009).

[9] Maxfield, C**.**, " The Design Warrior's Guide to FPGA: devices, tools and flows", ***Menthor Graphic Corporation and Xilinx, Inc****,* 2-51, 2004

[10] Küçükgüzel, E., "FPGA (Field Programmable Gate Array) lere Giriş, Kısa Tarihçe ve Örnek Uygulama", ***TOLA****,3-14, 2005*

[11] Betz, Vaughn, *FPGA Architecture for the Challenge*, http://www.eecg.toronto.edu/~vaughn/challenge/fpga_arch.html **(Ziyaret tarihi: 09 Kasım 2010).**

[12] Coffman, K., "Real World FPGA Design With Verilog", ***Prentice-Hall, Inc****,* 2001

[13] Zeidman, Bob, Mayıs 2006, *The Death of the Structured ASIC* [çevrimiçi], Chip Design Magazine, http://chipdesignmag.com/display.php?articleId=434&issueId=16 **(Ziyaret tarihi: 10 Ekim 2010)**.

[14] Aydın, A., "FPGA Yonga Mimarisi ve Kullanımı",Bitirme Projesi, ***Süleyman Demirel Üniversitesi Elektronik ve Haberleşme Mühendisliği,*** Isparta, 20-37 ,2005

[15] Bellis, M., 1997, Touch Screen[çevirimiçi], about.com, http://inventors.about.com/library/inventors/bltouch.htm **(27 Ağustos 2010)**

[16] Stumpe, B., 1977, CERN, "*A new principle for x-y touch system*", http://cdsweb.cern.ch/record/1266588/files/StumpeMar77.pdf, **(Ziyaret tarihi: 25 Mayıs 2010)**

[17] Beck F., Stumpe, B., CERN, "Two devices for operator interaction in the central control of the new CERN accelerator", http://cdsweb.cern.ch/record/ 186242/files/p1.pdf, **(Ziyaret tarihi: 25 Mayıs 2010)**

[18] Howstuffworks, 2008, "How do touch-screen monitors know where you're touching? [çevirimiçi]", Howstuffworks/Computer Monitors, http://computer. howstuffworks.com/question716.htm **(Ziyaret tarihi: 05 Ekim 2010)**.

[19] Touchscreens.com, "*Touch Screen Technologies"* ,.http://www.Touchscreens .com/intro-touchtypes.html [Çevirim Dışı], **(Ziyaret tarihi: 10 Nisan 2010)**.

[20] Xilinx, Inc., "EDK Concepts,Tools,and Techniques - *A Hands-On Guide to Effective Embedded System Design*", ***Xilinx, UG683 EDK 11***, 53-76, (2009).

[21] Freescale,"MC1322x- Advanced ZigBee™- Compliant SoC Platform for the 2.4 GHz IEEE® 802.15.4 Standard Reference Manual", ***Freescale, MC1322xRM***, 15-1,15-18,(2009)

[22] Wiki 2010, *Serial Peripheral Interface Bus* [çevirimiçi], http://en.wiki pedia.org/wiki/ Serial_Peripheral_Interface_Bus, **(Ziyaret tarihi: 10 Ekim 2010)**.

[23] Mct, 2000, *SPI - Serial Peripheral Interface* [çevirimiçi], http://www.mct.net/faq/spi.html **(Ziyaret tarihi: 2 Ekim 2010)**.

[24] Kalinsky D. & Kalinsky R. 2002, *Introduction to Serial Peripheral Interface*, http://www.embedded.com/story/OEG20020124S0116 [çevirim dışı], **(Ziyaret tarihi: 10 Mart 2010)**.

[25] ThaiEasyElec, *Breakout Board 2.8 inch QVGA TFT LCD with Touch Screen* [çevrimiçi], http://www.thaieasyelec.net/index.php/Display-Module/ Breakout-Board/Breakout-Board-2-8-inch-QVGA-TFT-LCD-with-Touch-Screen/p_3.html **(Ziyaret tarihi: 03 Mayıs 2010)**.

[26] Adobe Systems Incorporated, 2009, *Adobe AIR* [çevirimiçi], http://www.adobe.com/products/air/ **(Ziyaret tarihi: 12 02 2011)**.

[27] Adobe Systems Incorporated, 2009, *Adobe AIR Developer Center* [çevirimiçi], http://www.adobe.com/devnet/air.html **(Ziyaret tarihi: 12 02 2011)**

EKLER

EK-A: VHDL Tabanlı Kodlar

EK-A.1: TC_LCD_CORE VHDL Kodları

```
------------------------------------------------------------------------------
-- user_logic.vhd - entity/architecture bölümleri
------------------------------------------------------------------------------

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

--library proc_common_v3_00_a;
--use proc_common_v3_00_a.proc_common_pkg.all;

------------------------------------------------------------------------------
-- Entity bölümü
------------------------------------------------------------------------------
-- Definition of Generics:
--   C_SLV_DWIDTH              -- Slave interface data bus width
--   C_NUM_REG                 -- Number of software accessible registers
--
-- Definition of Ports:
--   Bus2IP_Clk                -- Bus to IP clock
--   Bus2IP_Reset              -- Bus to IP reset
--   Bus2IP_Addr               -- Bus to IP address bus
--   Bus2IP_Data               -- Bus to IP data bus
--   Bus2IP_BE                 -- Bus to IP byte enables
--   Bus2IP_RdCE               -- Bus to IP read chip enable
--   Bus2IP_WrCE               -- Bus to IP write chip enable
--   IP2Bus_Data               -- IP to Bus data bus
--   IP2Bus_RdAck              -- IP to Bus read transfer acknowledgement
--   IP2Bus_WrAck              -- IP to Bus write transfer acknowledgement
--   IP2Bus_Error              -- IP to Bus error response
------------------------------------------------------------------------------

entity user_logic is
  generic
  (
    C_SLV_DWIDTH                : integer              := 32;
```

```
  C_NUM_REG                    : integer              := 1
  );
  port
  (
    -- Kullanıcı giirş çıkışları başı ------------------
          lcd_cs                                                  : out  std_logic;
          lcd_rs                                                  : out  std_logic;
          lcd_wr                                                        :          out
std_logic;
          lcd_rd                                                  : out  std_logic;
          lcd_data                                                :                  out
std_logic_vector(0 to 7);
    -- Kullanıcı giirş çıkışları sonu ------------------

    -- Bus protocol ports, do not add to or delete
    Bus2IP_Clk                  : in  std_logic;
    Bus2IP_Reset                : in  std_logic;
    Bus2IP_Addr                 : in  std_logic_vector(0 to 31);
    Bus2IP_Data                 : in  std_logic_vector(0 to C_SLV_DWIDTH-1);
    Bus2IP_BE                   : in  std_logic_vector(0 to C_SLV_DWIDTH/8-1);
    Bus2IP_RdCE                 : in  std_logic_vector(0 to C_NUM_REG-1);
    Bus2IP_WrCE                 : in  std_logic_vector(0 to C_NUM_REG-1);
    IP2Bus_Data                 : out std_logic_vector(0 to C_SLV_DWIDTH-1);
    IP2Bus_RdAck                : out std_logic;
    IP2Bus_WrAck                : out std_logic;
    IP2Bus_Error                : out std_logic
  );

  attribute SIGIS : string;
  attribute SIGIS of Bus2IP_Clk    : signal is "CLK";
  attribute SIGIS of Bus2IP_Reset  : signal is "RST";

end entity user_logic;

------------------------------------------------------------------------------
-- Architecture bölümü
------------------------------------------------------------------------------

architecture IMP of user_logic is

  --USER signal declarations added here, as needed for user logic
signal count:              std_logic_vector (28 downto 0);
signal say :                                    std_logic_vector(7        downto        0)
:="00001100";
signal say2 :                                   std_logic_vector(7        downto        0)
:="00001100";
signal datah: std_logic_vector(0 to 7);
signal datal: std_logic_vector(0 to 7);
begin
```

```
clock_divider: process(BUS2IP_Clk)
   begin
      if (BUS2IP_Clk 'event and BUS2IP_Clk = '1') then
         count <= count + 1;
      end if;
end process clock_divider;


  --USER logic implementation added here
decode: process(BUS2IP_Clk, Bus2IP_Reset, Bus2IP_WrCE, Bus2IP_Addr(28 to
29))
   begin
       if (BUS2IP_Clk 'event and BUS2IP_Clk = '1') then
         --varsayilan ayarlar icin
                                if ((Bus2IP_Reset = '0') and (Bus2IP_WrCE(0) = '1')
and (Bus2IP_Addr(27 to 29) = "000")) then
         end if;
                                --write data high byte
         if ((Bus2IP_Reset = '0') and (Bus2IP_WrCE(0) = '1') and (Bus2IP_Addr(27
to 29) = "001")) then
                                        datah<= Bus2IP_Data(0 to 7);
                                        datal<= Bus2IP_Data(8 to 15);

                                        say<="00000000";
                                         lcd_cs <= '1';
                                         lcd_rs <= '0';
            lcd_wr <= '1';
            lcd_rd <= '1';
                                          --lcd_data(0 to 7)      <="00000000";
                                end if;

                                if(say="00000000") then
--s1. adım
                                lcd_rs <= '1';            --Set bit RS for data;
                                end if;

                                if(say="00000001") then --1
--s2. adım
                                lcd_cs <= '0';            --
                                lcd_wr <= '0';            --Write Enable
                                lcd_rd <= '1';            --Read Disable
                                lcd_data(0 to 7) <= datah;
                        end if;

                                if(say="00000100") then --4
--s3. adım
                                lcd_cs <= '1';--setBitCS;
                                lcd_wr <= '1';--setBitWR
```

```
                end if;

                        if(say="00000101") then --5
--s4. adım
                        lcd_rs <= '0';          --Clr bit RS for data;
                end if;
-- data low byte
                        if(say="00000110") then --6
--sL1. adım
                        lcd_rs <= '1';          --Set bit RS for data;
                        end if;

                        if(say="00000111") then --7
--sL2. adım
                        lcd_cs <= '0';          --
                        lcd_wr <= '0';          --Write Enable
                        lcd_rd <= '1';          --Read Disable
                        lcd_data(0 to 7) <= datal;
                end if;

                        if(say="00001010") then --10
--sL3. adım
                        lcd_cs <= '1';--setBitCS;
                        lcd_wr <= '1';--setBitWR
                end if;

                        if(say="00001011") then --11
--sL4. adım
                        lcd_rs <= '0';          --Clr bit RS for data;
                end if;

-- say++
        if (say<"00001100") then
                        say <= say + "00000001";
                        end if;


                        --write ins high byte
        if ((Bus2IP_Reset = '0') and (Bus2IP_WrCE(0) = '1') and (Bus2IP_Addr(27
to 29) = "011")) then
                                datah<= Bus2IP_Data(0 to 7);
                                datal<= Bus2IP_Data(8 to 15);
                                say2<="00000000";
                                 lcd_cs <= '1';
                                 lcd_rs <= '1';
            lcd_wr <= '1';
            lcd_rd <= '1';
                        end if;
```

```
                        if(say2="00000000") then
--s1. adım
                        lcd_rs <= '0';          --Set bit RS for data;
                        end if;

                        if(say2="00000001") then --1
--s2. adım
                        lcd_cs <= '0';          --
                        lcd_wr <= '0';          --Write Enable
                        lcd_rd <= '1';          --Read Disable
                        lcd_data(0 to 7) <= datah;
                end if;

                        if(say2="00000100") then --4
--s3. adım
                        lcd_cs <= '1';--setBitCS;
                        lcd_wr <= '1';--setBitWR
                end if;

                        if(say2="00000101") then --5
--s4. adım
                        lcd_rs <= '1';          --Clr bit RS for data;
                end if;

-- data low byte
                        if(say2="00000110") then --6
--sL1. adım
                        lcd_rs <= '0';          --Set bit RS for data;
                        end if;

                        if(say2="00000111") then --7
--sL2. adım
                        lcd_cs <= '0';          --
                        lcd_wr <= '0';          --Write Enable
                        lcd_rd <= '1';          --Read Disable
                        lcd_data(0 to 7) <= datal;
                end if;

                        if(say="00001010") then --10
--sL3. adım
                        lcd_cs <= '1';--setBitCS;
                        lcd_wr <= '1';--setBitWR
                end if;

                        if(say="00001011") then --11
--sL4. adım
                        lcd_rs <= '1';          --Clr bit RS for data;
                end if;
```

```
-- say2++
          if (say2<"00001100") then
                                    say2 <= say2 + "00000001";
                                    end if;



      end if;
end process decode;

  ----------------------------------------
  -- Example code to drive IP to Bus signals
  ----------------------------------------
--IP2Bus_Data <= (others => '0');

  IP2Bus_WrAck <= Bus2IP_WrCE(0);
  IP2Bus_RdAck <= Bus2IP_RdCE(0);
  IP2Bus_Error <= '0';
end IMP;
```

EK-A.2: SPI_CORE VHDL Kodları

```
------------------------------------------------------------------------------
-- user_logic.vhd - entity/architecture bölümleri
------------------------------------------------------------------------------

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

library proc_common_v3_00_a;
use proc_common_v3_00_a.proc_common_pkg.all;



------------------------------------------------------------------------------
-- Entity bölümü
------------------------------------------------------------------------------
-- Definition of Generics:
--   C_SLV_DWIDTH              -- Slave interface data bus width
--   C_NUM_REG                 -- Number of software accessible registers
--
-- Definition of Ports:
--   Bus2IP_Clk                -- Bus to IP clock
--   Bus2IP_Reset              -- Bus to IP reset
--   Bus2IP_Addr               -- Bus to IP address bus
--   Bus2IP_Data               -- Bus to IP data bus
--   Bus2IP_BE                 -- Bus to IP byte enables
--   Bus2IP_RdCE               -- Bus to IP read chip enable
--   Bus2IP_WrCE               -- Bus to IP write chip enable
--   IP2Bus_Data               -- IP to Bus data bus
--   IP2Bus_RdAck              -- IP to Bus read transfer acknowledgement
--   IP2Bus_WrAck              -- IP to Bus write transfer acknowledgement
--   IP2Bus_Error              -- IP to Bus error response
------------------------------------------------------------------------------

entity user_logic is
  generic
  (
    C_SLV_DWIDTH                : integer              := 32;
    C_NUM_REG                   : integer              := 1
  );
  port
  (
    -- Kullanıcı giirş çıkışları başı ------------------
                              sd_SS:          out                 std_logic;
                              sd_MOSI: out            std_logic;
```

```
                        sd_MISO: in                 std_logic;
                        sd_SCK:     out             std_logic;
                        sd_DAT1: out        std_logic;
                        sd_DAT2: out        std_logic;
                        sd_CD:      in                  std_logic;
                        sd_WP:      in                  std_logic;
    -- Kullanıcı giirş çıkışları sonu ------------------

    -- Bus protocol ports, do not add to or delete
    Bus2IP_Clk              : in  std_logic;
    Bus2IP_Reset            : in  std_logic;
    Bus2IP_Addr             : in  std_logic_vector(0 to 31);
    Bus2IP_Data             : in  std_logic_vector(0 to C_SLV_DWIDTH-1);
    Bus2IP_BE               : in  std_logic_vector(0 to C_SLV_DWIDTH/8-1);
    Bus2IP_RdCE             : in  std_logic_vector(0 to C_NUM_REG-1);
    Bus2IP_WrCE             : in  std_logic_vector(0 to C_NUM_REG-1);
    IP2Bus_Data             : out std_logic_vector(0 to C_SLV_DWIDTH-1);
    IP2Bus_RdAck            : out std_logic;
    IP2Bus_WrAck            : out std_logic;
    IP2Bus_Error            : out std_logic

  );

  attribute SIGIS : string;
  attribute SIGIS of Bus2IP_Clk    : signal is "CLK";
  attribute SIGIS of Bus2IP_Reset  : signal is "RST";

end entity user_logic;

------------------------------------------------------------------------------
-- Architecture bölümü
------------------------------------------------------------------------------

architecture IMP of user_logic is


--BUS_DATA
signal BUS_DATA                                 : STD_LOGIC_VECTOR (0 to
31);

--SPI_DATA
signal SPI_DATA                                     : STD_LOGIC_VECTOR
(0 to 31);
signal SPI_TX_DATA                                  : STD_LOGIC_VECTOR
(0 to 31);
signal SPI_RX_DATA                                  : STD_LOGIC_VECTOR
(0 to 31);
```

```
--SPI_CLK_CTRL
signal SPI_CLK_CTRL                    : STD_LOGIC_VECTOR (0 to
31);

signal SPI_START                       : STD_LOGIC;
                              --SPI_CLK_CTRL(7)
signal SPI_SCK_COUNT                   : STD_LOGIC_VECTOR (0 to 7); --
SPI_CLK_CTRL(15 downto 8)
signal SPI_SCK_DATA_LENGTH             : STD_LOGIC_VECTOR (0 to 6); --
SPI_CLK_CTRL(6 downto 0)


--SPI_SETUP
signal SPI_SETUP                       : STD_LOGIC_VECTOR (0 to
31);

signal SPI_SCK_FREQ                    : STD_LOGIC_VECTOR (0 to 2);   --
SPI_SETUP(14 downto 12)
signal SPI_MISO_PHASE                  : STD_LOGIC;
                        --SPI_SETUP(10)
signal SPI_SCK_PHASE                   : STD_LOGIC;
                        --SPI_SETUP(9)
signal SPI_SCK_POL                     : STD_LOGIC;
                        --SPI_SETUP(8)
signal SPI_3WIRE                       : STD_LOGIC;
                        --SPI_SETUP(17)
signal SPI_MODE                        : STD_LOGIC;
                        --SPI_SETUP(16)
signal SPI_SDO_INACTIVE_ST     : STD_LOGIC_VECTOR (0 to 1);   --
SPI_SETUP(5 downto 4)
signal SPI_SS_DELAY                    : STD_LOGIC_VECTOR (0 to 1);   --
SPI_SETUP(3 downto 2)
signal SPI_SS_SETUP                    : STD_LOGIC_VECTOR (0 to 1);   --
SPI_SETUP(1 downto 0)

signal SPI_SETUP_START                 : STD_LOGIC:='1';


--clk degislenleri
signal p_clk    : std_logic;              -- secilmis clock
signal p1_clk   : std_logic:=SPI_SETUP(9);
signal p0_clk   : std_logic:=SPI_SETUP(8);
signal div_clk : std_logic;               -- bolunmus clk
signal ref_clk : std_logic:='0';          -- referans clk
signal dat2     : std_logic;     -- gecici 10 clk



--diger degiskenler
```

```
signal count:                std_logic_vector (7 downto 0);
signal data_rx:                    integer :=11;
signal reg_interrupt:         std_logic:='1';
signal p_clk_count:           std_logic_vector (7 downto 0):="00000000";
signal bekle_count:           std_logic_vector (2 downto 0):="000";
signal durum:                      std_logic_vector (2 downto 0):="101";
signal veri_hazir:                        std_logic:='0';
signal adim:                                    std_logic:='0';
signal adim_at:                           std_logic:='0';
signal p_clk_onceki_durum:          std_logic:='0';
signal p_clk_durum:                     std_logic:='0';

signal sayac:                                              std_logic_vector (31 downto
0):="00000000000000000000000000000000";

begin

spiSetup: process(BUS2IP_Clk)
   begin



                        -- SPI_SETUP
                        SPI_3WIRE                                  <=
SPI_SETUP(17);
                        SPI_MODE                          <= SPI_SETUP(16); --
spi_mode is low SPI -> master mode high SPI -> slave mode
                        SPI_SCK_FREQ                      <= SPI_SETUP(12 to 14);
                        SPI_MISO_PHASE                    <= SPI_SETUP(10);
                        SPI_SCK_PHASE                     <= SPI_SETUP(9);
                        SPI_SCK_POL                       <= SPI_SETUP(8);
                        SPI_SDO_INACTIVE_ST               <= SPI_SETUP(4 to 5);
                        SPI_SS_DELAY                      <= SPI_SETUP(2 to 3);
                        SPI_SS_SETUP                      <= SPI_SETUP(0 to 1);

                        -- SPI_CLK_CTRL
                        SPI_START                         <= SPI_CLK_CTRL(7);
                        SPI_SCK_COUNT                     <= SPI_CLK_CTRL(8 to
15);
                        SPI_SCK_DATA_LENGTH                    <=
SPI_CLK_CTRL(0 to 6);

end process spiSetup;

main: process (BUS2IP_Clk, Bus2IP_Reset, Bus2IP_WrCE, Bus2IP_Addr(28 to
29))
                begin
                if (BUS2IP_Clk 'event and BUS2IP_Clk='1') then
                        --SPI_TX_DATA
```

```
                    if ((Bus2IP_Reset = '0') and (Bus2IP_WrCE(0) = '1') and
(Bus2IP_Addr(27 to 29) = "000")) then
                                SPI_TX_DATA <= Bus2IP_Data ;
                    end if;
                    --SPI_RX_DATA
                    if ((Bus2IP_Reset = '0') and (Bus2IP_WrCE(0) = '1') and
(Bus2IP_Addr(27 to 29) = "001")) then
                                BUS_DATA <= SPI_RX_DATA ;
                    end if;

                    --SPI_CLK_CTRL
                    if ((Bus2IP_Reset = '0') and (Bus2IP_WrCE(0) = '1') and
(Bus2IP_Addr(27 to 29) = "010")) then
                                SPI_CLK_CTRL <= Bus2IP_Data ;
                    end if;
                    --SPI_SETUP
                    if ((Bus2IP_Reset = '0') and (Bus2IP_WrCE(0) = '1') and
(Bus2IP_Addr(27 to 29) = "011")) then
                                SPI_SETUP <= Bus2IP_Data ;
                    end if;

                    -- SPI_STATUS
                    if ((Bus2IP_Reset = '0') and (Bus2IP_WrCE(0) = '1') and
(Bus2IP_Addr(27 to 29) = "100")) then
                                BUS_DATA
<="11110000111100000000111100000"& durum;
                    end if;

                    if ((Bus2IP_Reset = '0') and (Bus2IP_WrCE(0) = '1') and
(Bus2IP_Addr(27 to 29) = "101")) then
                                durum <="001";
                    end if;


                        if (durum<"101") then
                                -- ilk bekleme delay+1 yap
                                if (durum="010" or durum="100") then
                                        if
(bekle_count<=SPI_SS_DELAY) then
                                                if      (adim_at='1')
then

        bekle_count <= bekle_count + "001";
                                                end if;
                                        end if;
                                        if      (bekle_count      >
SPI_SS_DELAY) then

        bekle_count<="000";
```

```
        durum<=durum+"001";
                                                    end if;
                                                elsif (durum="011") then
                                                    if      (p_clk_count      >
SPI_SCK_COUNT) then

        durum<=durum+"001";
                                                    end if;

                                                else
                                                        durum<=durum+"001";
                                                end if;
                                    end if;
                        end if; --BUS2IP_Clk end if;
end process main;

main_DURUM:         process (BUS2IP_Clk)
                    begin
                    if (BUS2IP_Clk 'event and BUS2IP_Clk='1') then

                                    count <= count + 1;
                                    if (SPI_SCK_FREQ="000")  then
        div_clk<=count(0); end if;
                                    if (SPI_SCK_FREQ="001")  then
        div_clk<=count(1); end if;
                                    if (SPI_SCK_FREQ="010")  then
        div_clk<=count(2); end if;
                                    if (SPI_SCK_FREQ="011")  then
        div_clk<=count(3); end if;
                                    if (SPI_SCK_FREQ="100")  then
        div_clk<=count(4); end if;
                                    if (SPI_SCK_FREQ="101")  then
        div_clk<=count(5); end if;
                                    if (SPI_SCK_FREQ="110")  then
        div_clk<=count(6); end if;
                                    if (SPI_SCK_FREQ="111")  then
        div_clk<=count(7); end if;


                                    --p_clk_durum <-- p_clk clk vermeye ortadan degilde
tam bastan baslamasi icin

                                    --if             (p_clk>p_clk_onceki_durum             or
p_clk<p_clk_onceki_durum) then
                                            if (div_clk=p_clk) then p_clk_durum <= '1'; --
end if;
                                            --p_clk_durum <= '1';
                                    else
```

```
                        p_clk_durum <= '0';
                end if;


                p_clk_onceki_durum <= p_clk;


                if (veri_hazir='1' and p_clk_durum='1')       then
                                                        -- sck gonder

                                                        sd_SCK <= p_clk;
        -- secilmis clk
                end if;



                if (div_clk=adim) then
                        adim_at<='0';
                else
                                adim<=div_clk;
                        if (div_clk='1') then
                                adim_at<='1';
                        end if;
                end if;

                        if (durum="001") then

                                --SPI_SCK_PHASE                 <=
SPI_SETUP(9);
                                --SPI_SCK_POL                   <=
SPI_SETUP(8);
                                sd_SCK <= SPI_SETUP(8); --      bittigi
gibi baslasin
                                SPI_DATA<= SPI_TX_DATA;
                                p_clk_count <="00000000";
                                veri_hazir<='0';
                                if (SPI_START='1') then
                                        sd_SS  <='1'; -- sd_SS ilk deger
                                        reg_interrupt<='1';
                                end if;

                        end if;
                        if (durum="010") then
                                sd_SS <='0'; -- sd_SS start
                                -- ilk bekleme noktasi (delay+1)

                                sd_DAT1 <='1';
                                veri_hazir<='0';
                        end if;
```

```
                                        if (durum="011") then
                                                if (adim_at='1') then
                                                        if (ref_clk='1') then ref_clk<='0';
else ref_clk<='1'; end if;
                                                                if (ref_clk='1') then
                                                                        sd_DAT1 <='0';
                                                                        if
(p_clk_count<=SPI_SCK_COUNT) then
                                                                                --
SPI_SCK_DATA_LENGTH'e gore reg gonder al
                                                                                --      data
gonder -- bu ref clk ya gore gidecek
                                                                                sd_MOSI
                                <= SPI_DATA(0);     -- sd_out

        SPI_DATA<=   SPI_DATA(1 to 31)  & sd_MISO;

        veri_hazir<='1';
                                                                                --      sayac
artir

        p_clk_count <= p_clk_count + "00000001";
                                                                        end if;
                                                                end if;
                                                end if;
                                        end if;
                                        if (durum="100") then
                                                veri_hazir<='0';
                                                sd_DAT2 <='1';
                                                -- son bekleme noktasi (delay+1)
                                        end if;
                                        if (durum="101") then
                                                sd_DAT2 <='0';
                                                sd_SS  <='1'; -- sd_SS dur
                                                -- reg_interrrupt 0 gelen veri hazir
                                                reg_interrupt<='0';
                                                SPI_RX_DATA <= SPI_DATA;
                                                --SPI_SCK_PHASE                 <=
SPI_SETUP(9);
                                                --SPI_SCK_POL                   <=
SPI_SETUP(8);
                                                sd_SCK <= SPI_SETUP(8);  --      saat
bittigi gibi baslamasi icin

                                        end if;
                        end if;
end process main_DURUM;
```

```
refClk: process (div_clk)
                begin

                if (div_clk 'event and div_clk='1' and SPI_SCK_PHASE='1') then
                        if (SPI_SCK_POL='1') then
                                if (p1_clk='1') then  p1_clk <='0';
                                                        else        p1_clk
<='1';
                                end if;
                        end if;
                        if (SPI_SCK_POL='0') then
                                if (p1_clk='0') then  p1_clk <='1';
                                                        else        p1_clk
<='0';
                                end if;
                        end if;
                end if;

                if (div_clk 'event and div_clk='0' and SPI_SCK_PHASE='0') then
                        if (SPI_SCK_POL='1') then
                                if (p0_clk='1') then  p0_clk <='0';
                                                        else        p0_clk
<='1';
                                end if;
                        end if;
                        if (SPI_SCK_POL='0') then
                                if (p0_clk='0') then  p0_clk <='1';
                                                        else        p0_clk
<='0';
                                end if;
                        end if;
                end if;


                if (SPI_SCK_PHASE='1') then
                        p_clk<= p1_clk;
                end if;
                if (SPI_SCK_PHASE='0') then
                        p_clk<= p0_clk;
                end if;

end process refClk;


-- Hazirlanan verinin veryoluna yazilmasi
BUStoIP_DATA:       process (BUS_DATA)
                begin
                        if (BUS2IP_Clk 'event and BUS2IP_Clk='1') then
```

```
                IP2Bus_Data <= BUS_DATA;

                IP2Bus_WrAck <= Bus2IP_WrCE(0);
                IP2Bus_RdAck <= Bus2IP_RdCE(0);
                IP2Bus_Error <= '0';
            end if;

    end process BUStoIP_DATA;


    end IMP;
```

EK-B: C Tabanlı Kodlar

EK-B.1: Ana Program (Main.c) CKodları.

```
//----------------------------------------
//        Header Files
//----------------------------------------
#include "xparameters.h"
#include "stdio.h"
#include "xbasic_types.h"
#include "xgpio.h"
#include "lcd240320tp.h"
#include "resimler/smile.h"


#include "mb_interface.h"
#include "xutil.h"
#include "stdlib.h"
#include "string.h"



#define UARTLITE_BASEADDR       XPAR_UARTLITE_0_BASEADDR

//----------------------------------------
//        Global Variables
//----------------------------------------


XGpio GpioOutRST;           /* The driver instance for GPIO Device configured as
O/P */
XGpio GpioOutBL;    /* The driver instance for GPIO Device configured as O/P */


//----------------------------------------------------------------------
//        MAIN
//----------------------------------------------------------------------
int main (void) {


  int *addr_ptr;
  Xuint16 renkKodu,renkKodu1,renkKodu2,renkKodu3,renkKodu4,renkKodu5;
  Xuint8 recv;

  char f[100];
  char f_g[1];
  char
f_adi[40],parametre_1[40],parametre_2[40],parametre_3[40],parametre_4[40],param
etre_5[40];
```

```
  int i=0,j=0,k=0,l=0,m=0,n=0,p=0,r=0,s=0,p1=0,p2=0,p3=0,p4=0,p5=0,p6=0;
  int par1toSay1=0,par2toSay2=0,par3toSay3=0,par4toSay4=0,par5toSay5=0; //print
str de cok basamakli sayilar icin
  int f_yuz_say=0;
  recv=0;

          //sayi icin
          int     basamak;
          int     basamak_sayisi;
          int     basamak_atla;
          //renk icin
          int     renk_kaydir;
          word renk_kodu1,renk_kodu4,renk_kodu5;
          char renk_1;
          //res adres icin
          Xuint32 resAdres;
          int res_adres_kaydir=0;
          char res_adres_1=0;


   /* Enable and initialize cache */
   #if XPAR_MICROBLAZE_0_USE_ICACHE
     microblaze_invalidate_icache();
     microblaze_enable_icache();
   #endif

   #if XPAR_MICROBLAZE_0_USE_DCACHE
     microblaze_invalidate_dcache();
      microblaze_enable_dcache();
   #endif

   print("\r\n TCLCD hazirlaniyor...\r\n");


          initGpIO(); //giris cikis birimlerini hazir hale getir
          TSLCDRst();
          TSLCDInit();

          delay(3000);


//****_____*****\\



  xil_printf ("\r\n\r\n* --> FPGA' ya Sistem basariyla yuklendi. <-- *\r\n");
  xil_printf ("* --> Bir Fonksiyon Giriniz ve 'Enter' Tusuna Basiniz.  *\r\n");
```

```
  while (recv!=24) { //esc gelmediyse devam et


/* Read the keyboard, print the entry.  */
          recv = XUartLite_RecvByte(UARTLITE_BASEADDR);
          f[f_yuz_say]=recv;
          f_yuz_say++;




          if (recv==59)  // ";" geldiyse buraya girer
          {


////////////////////////////////////////////////////////////
//gelen veri diziye "f" alinir
////////////////////////////////////////////////////////////


            for (i=0;i<strlen(f);i++) {
        //
          f_g[0]=f[i];
          f_g[1]='\0';
          //printf("--> %S \r\n", f_g);


          if(p2==1) {
                        parametre_2[m]=f[i];
                        m++;
                        }
          if(p3==1) {
                        parametre_3[n]=f[i];
                        n++;
                        }
          if(p4==1) {
                        parametre_4[p]=f[i];
                        p++;
                        }
          if(p5==1) {
                        parametre_5[r]=f[i];
                        r++;
                        }
                              if(strncmp(f_g,")",1)==0) {
             p1=0;
             p2=0;
             p3=0;
             p4=0;
             p5=0;
```

```
            p6=0;

        }
        if(strncmp(f_g,",",1)==0) {

                if (p5==1) {
                   p5=0;p6=1;
                }
                if (p4==1) {
                   p4=0;p5=1;
                }
                if (p3==1) {
                   p3=0;p4=1;
                }
                if (p2==1) {
                   p2=0;p3=1;
                }
                if (p1==1) {
                   p1=0;p2=1;
                }
        }
        if(p1==1) {
                        parametre_1[l]=f[i];
                        l++;
                        }
        if(strncmp(f_g,"(",1)==0) {
                p1=1;      // parametreleri atama yapmaya başla
                strncpy (f_adi,f,i);
        }

    }
          //xil_printf ("ALINAN deger0 !!!  %s  !!!!\r\n", f_adi);

    parametre_1[l]='\0';
    //xil_printf ("ALINAN deger1(str) !!!  %d  !!!!\r\n", parametre_1);
          ////////////////////
                  // To Int
                                          basamak=1;
                                          basamak_sayisi=l;
                                          basamak_atla=1;
                                          renk_kaydir=0;
                                          renk_1=0;
                                          renk_kodu1=0x0000;

                                          l--;
                                          while (basamak_sayisi>=basamak){
                                                  //xil_printf ("par_1_deger--  %d
!!!!\r\n", parametre_1[l]-48 );
```

```
                                        //xil_printf ("basamak--  %d  !!!!\r\n",
basamak_atla );

        par1toSay1=par1toSay1+((parametre_1[l]-48)*basamak_atla);
                                        basamak_atla=basamak_atla*10;
                                        //xil_printf ("||||parsay !!!  %d  !!!!\r\n",
par1toSay1 );
                                        basamak++;
                                         //renk kodu 1
                                        if (renk_kaydir<=12) {
                                                renk_1=parametre_1[l]-48; if
(renk_1>9) renk_1=renk_1-7;
                                                renk_kodu1    |=
renk_1<<renk_kaydir;
                                                renk_kaydir+=4;
                                                }

                                        l--;

                                }
                                                //xil_printf ("--> %x
\r\n",renk_kodu1);


   //xil_printf ("ALINAN deger1(int) !!!  %d  !!!!\r\n", par1toSay1);
        ////////////////////

   m--;
   parametre_2[m]='\0';
   //xil_printf ("ALINAN deger2(str) !!!  %d  !!!!\r\n", parametre_2);
        ////////////////////
                // To Int
                                basamak=1;
                                basamak_sayisi=m;
                                basamak_atla=1;

                                m--;
                                while (basamak_sayisi>=basamak){
                                        //xil_printf ("par_2_deger--  %d
!!!!\r\n", parametre_2[m]-48 );
                                        //xil_printf ("basamak--  %d  !!!!\r\n",
basamak_atla );

        par2toSay2=par2toSay2+((parametre_2[m]-48)*basamak_atla);
                                        basamak_atla=basamak_atla*10;
                                        //xil_printf ("||||parsay !!!  %d  !!!!\r\n",
par2toSay2 );
                                        basamak++;
                                        m--;
```

```
                                        }
    //xil_printf ("ALINAN deger2(int) !!!  %d  !!!!\r\n", par2toSay2);
            ////////////////////

    n--;
    parametre_3[n]='\0';
    //xil_printf ("ALINAN deger3 !!!  %s  !!!!\r\n", parametre_3);
            ////////////////////
                    // To Int
                                        basamak=1;
                                        basamak_sayisi=n;
                                        basamak_atla=1;
                                        n--;
                                        while (basamak_sayisi>=basamak){

            par3toSay3=par3toSay3+((parametre_3[n]-48)*basamak_atla);
                                                basamak_atla=basamak_atla*10;
                                                basamak++;
                                                n--;
                                        }

    //xil_printf ("ALINAN deger3(int) !!!  %d  !!!!\r\n", par3toSay3);
            ////////////////////

    p--;
    parametre_4[p]='\0';
    //xil_printf ("ALINAN deger4 !!!  %s  !!!!\r\n", parametre_4);
            ////////////////////
                    // To Int
                                        basamak=1;
                                        basamak_sayisi=p;
                                        basamak_atla=1;

                                        renk_kaydir=0;
                                        renk_1=0;
                                        renk_kodu4=0x0000;

                                        p--;
                                        while (basamak_sayisi>=basamak){

            par4toSay4=par4toSay4+((parametre_4[p]-48)*basamak_atla);
                                                basamak_atla=basamak_atla*10;
                                                basamak++;
                                                 //renk kodu 4
                                                if (renk_kaydir<=12) {
                                                        renk_1=parametre_4[p]-48; if
(renk_1>9) renk_1=renk_1-7;
                                                        renk_kodu4    |=
renk_1<<renk_kaydir;
```

```
                                                renk_kaydir+=4;
                                                }
                                        p--;
                                }
   //xil_printf ("ALINAN deger4(int) !!!  %d  !!!!\r\n", par4toSay4);
        ////////////////////

   r--;
   parametre_5[r]='\0';
   //xil_printf ("ALINAN deger5 !!!  %s  !!!!\r\n", parametre_5);
        ///////////////////
                // To Int
                                basamak=1;
                                basamak_sayisi=r;
                                basamak_atla=1;
                                renk_kaydir=0;
                                renk_1=0;
                                renk_kodu5=0x0000;

                                res_adres_kaydir=0;
                                res_adres_1=0;
                                resAdres=0x00000000;

                                r--;
                                while (basamak_sayisi>=basamak){

        par5toSay5=par5toSay5+((parametre_5[r]-48)*basamak_atla);
                                        basamak_atla=basamak_atla*10;
                                        basamak++;
                                         //renk kodu 5
                                        if (renk_kaydir<=12) {
                                                renk_1=parametre_5[r]-48; if
(renk_1>9) renk_1=renk_1-7;
                                                renk_kodu5    |=
renk_1<<renk_kaydir;
                                                renk_kaydir+=4;
                                                }
                                         //resAdres 5
                                        if (res_adres_kaydir<=28) {
                                                res_adres_1=parametre_5[r]-48;
if (res_adres_1>9) res_adres_1=res_adres_1-7;
                                                resAdres        |=
res_adres_1<<res_adres_kaydir;
                                                res_adres_kaydir+=4;
                                                }

                                        r--;
                                }
```

```
   //xil_printf ("ALINAN deger5(int) !!!  %d  !!!!\r\n", par5toSay5);


//
//Yazi
if (strncmp(f_adi,"PrintStr",8)==0) {
        //xil_printf  ("Yazi Fonk--> %s(%d,%d,\"%s\",%s);
\r\n",f_adi,par1toSay1,par2toSay2,parametre_3,parametre_4);
  PrintStr(par1toSay1,par2toSay2,parametre_3);
        }
//Dikdotgen
if (strncmp(f_adi,"FillRect",8)==0) {
                //xil_printf  ("Dikdortgen Fonk--> %s(%d,%d,%s,%s,%s);
\r\n",f_adi,par1toSay1,par2toSay2,parametre_3,parametre_4,parametre_5);
                FillRect(par1toSay1,par2toSay2,par3toSay3,par4toSay4,renk_kodu5);
        }
//Çember Çizimi
if (strncmp(f_adi,"FillCirc",8)==0) {
        FillCirc(par1toSay1,par2toSay2,par3toSay3,renk_kodu4);
        }
//Arkaplan
if (strncmp(f_adi,"SetBackGroundColor",13)==0) {
        //xil_printf  ("Arka Plan Renk Fonk--> %s(%s); \r\n",f_adi,parametre_1);
        SetBackGroundColor(renk_kodu1);
        }

//Yazı Rengi Belirleme
if (strncmp(f_adi,"SetFontColor",12)==0)
SetFontColor(renk_kodu1);

//Yazı Arkaplan Rengi Belirleme
if (strncmp(f_adi,"SetBackColor",12)==0)
SetBackColor(renk_kodu1);

//Karakter Yazdırma
if (strncmp(f_adi,"CharDisp",8)==0)
CharDisp(par1toSay1,par2toSay2,parametre_3);

//Resim Ekleme
if (strncmp(f_adi,"ShowPic",7)==0)
        ShowPic(par1toSay1,par2toSay2,par3toSay3,par4toSay4,smile);

//Resim Ekleme
if (strncmp(f_adi,"ShowPicRAM",7)==0) {
        ShowPicRAM(par1toSay1,par2toSay2,par3toSay3,par4toSay4,resAdres);
        xil_printf  ("\r\n %d %d %d %d %x
\r\n",par1toSay1,par2toSay2,par3toSay3,par4toSay4,resAdres);
}
```

```
		// tum degiskenler burada sifirlanacak
		while(f_yuz_say>0){
			f[f_yuz_say]="\0";
			f_yuz_say--;
			}
			xil_printf  ("\r\nFonksiyon Isletildi\r\n");



		 //f[100];
			f_g[0]="\0";
			f_g[1]="\0";
			basamak_atla=1;
	  //int
f_adi[40],parametre_1[40],parametre_2[40],parametre_3[40],parametre_4[40],param
etre_5[40];

		i=0;j=0;k=0;l=0;m=0;n=0;p=0;r=0;s=0;p1=0;p2=0;p3=0;p4=0;p5=0;p6=0;

		par1toSay1=0;par2toSay2=0;par3toSay3=0;par4toSay4=0;par5toSay5=0;

			recv=0;
			f_yuz_say=0;


		}	//Enter sonu
}	//ESC sonu


//
//******_____**********\\


  /*
   * Disable cache and reinitialize it so that other
   * applications can be run with no problems
   */
  #if XPAR_MICROBLAZE_0_USE_DCACHE
    microblaze_disable_dcache();
    microblaze_invalidate_dcache();
  #endif

  #if XPAR_MICROBLAZE_0_USE_ICACHE
    microblaze_disable_icache();
    microblaze_invalidate_icache();
  #endif
```

```
  print("-- SON main() --\r\n");
  return 0;
}
```

EK-B.2: Dokunmatik Panel (TC240320TP.c) C Kodları.

```
#include "xparameters.h"
#include "mb_interface.h"
#include "stdio.h"
#include "xutil.h"
#include "stdlib.h"

  Xuint32 *addr_ptr;
  Xuint32 SPI_RX_DATA;
  Xuint32 SPI_STATUS;
  Xuint32 SPI_MB_DATA;

void touchControlSet(void)  //Baslangic Ayarlari
        {
/////###############Baslangic Ayarlari########\\\\\\\

                /* SPI_CLK_CTRL */
                addr_ptr = XPAR_TOUCHCONTROL_SPI_0_BASEADDR+8;
                        *addr_ptr = 0x19180000;

                /* SPI_SETUP */
                addr_ptr = XPAR_TOUCHCONTROL_SPI_0_BASEADDR+12;
                        *addr_ptr =     0x302C8000;
}

Xuint32 getTouchControlX(void) {

/* X */
                addr_ptr = XPAR_TOUCHCONTROL_SPI_0_BASEADDR;
                        *addr_ptr = 0XD0000000;

                addr_ptr = XPAR_TOUCHCONTROL_SPI_0_BASEADDR+20;
                        *addr_ptr = 0X00000000;

                        addr_ptr =
XPAR_TOUCHCONTROL_SPI_0_BASEADDR+4;
                        *addr_ptr = 0X00000000;
                        addr_ptr =
XPAR_TOUCHCONTROL_SPI_0_BASEADDR+4;
                        SPI_RX_DATA= *addr_ptr/0xF0;
                  return SPI_RX_DATA;
}


Xuint32 getTouchControlY(void) {

/* y */
                addr_ptr = XPAR_TOUCHCONTROL_SPI_0_BASEADDR;
```

```
                        *addr_ptr = 0X90000000;

                addr_ptr = XPAR_TOUCHCONTROL_SPI_0_BASEADDR+20;
                        *addr_ptr = 0X00000000;

                        addr_ptr =
XPAR_TOUCHCONTROL_SPI_0_BASEADDR+4;
                        *addr_ptr = 0X00000000;
                        addr_ptr =
XPAR_TOUCHCONTROL_SPI_0_BASEADDR+4;
                        SPI_RX_DATA= *addr_ptr/0x111;
                  return SPI_RX_DATA;
}

//
//
//              /* SPI_STATUS */
//              //addr_ptr = XPAR_TOUCHCONTROL_SPI_0_BASEADDR+16;
//              //      *addr_ptr = 0x00000000;
//              //addr_ptr = XPAR_TOUCHCONTROL_SPI_0_BASEADDR+16;
//              //      SPI_STATUS= *addr_ptr;
//              //   xil_printf (" SPI_STATUS= %x \r\n", SPI_STATUS);
//
//              /* DURUM=001 */
//              //      addr_ptr =
XPAR_TOUCHCONTROL_SPI_0_BASEADDR+20;
//              //      *addr_ptr = 0X00000000;
```

EK-B.3: LCD Ekran (LCD240320TP.c) C Kodları.

```
//----------------------------------------
//        Header Files
//----------------------------------------
#include "xparameters.h"
#include "xgpio.h"
#include "lcd240320tp.h"
#include "font/font_courier.h"
#include "font/font_vrinda11x16.h"
#include "font/font_vrinda18x16.h"
#include "mb_interface.h"
#include "stdio.h"
#include "xutil.h"
#include "stdlib.h"


//----------------------------------------
//        Global Variables
//----------------------------------------
ts_pos_t offsetx,offsety;
ts_pos_t ts_margin_xl = 0;
ts_pos_t ts_margin_xr = TS_SIZE_X - 1;
ts_pos_t ts_margin_yu = 0;
ts_pos_t ts_margin_yl = TS_SIZE_Y - 1;

int font_type;
word font_color;
word back_color;
byte char_buf[FONT_BIT_WIDTH][FONT_BYTE_HIGHT];


/* The following are declared globally so they are zeroed and so they are easily
accessible from a debugger*/
XGpio GpioOutRST;           /* The driver instance for GPIO Device configured as
O/P */
XGpio GpioOutBL;    /* The driver instance for GPIO Device configured as O/P */
#define CHANNEL 1           /* which channel of the GPIO is */


//----------------------------------------
//        Function Prototypes //GPIO Functions
//----------------------------------------
//XStatus GpioOutput(Xuint16 DataWrite);
void initGpIO();


//-------------------------------------------------------
// TUM FONKSIYONLAR-----------------------------------
//-------------------------------------------------
```

```
//-----------------------------------------
//        GpIO Initialization Function
//-----------------------------------------
void initGpIO()
{
        XGpio_Initialize(&GpioOutRST, XPAR_LCD_RST_DEVICE_ID);
  XGpio_SetDataDirection(&GpioOutRST, CHANNEL, 0x0);

        XGpio_Initialize(&GpioOutBL, XPAR_LCD_BL_DEVICE_ID);
  XGpio_SetDataDirection(&GpioOutBL, CHANNEL, 0x0);


        setBitRST;
        setBitBL;

}


//-------------------------------------
//        OUT DATA Function
//-------------------------------------
 Xuint16 *addr_ptr;
 int LED_value = 0;

void TSLCDOutDat(word dat)
{


        //LCD_W_Data = dat>>8;     //write data high byte
        //GpioOutput(dat>>8);
          addr_ptr = XPAR_LCD_CONTROL_0_BASEADDR + 4;
          *addr_ptr = dat ;

}

//-------------------------------------
//        OUT INSTRUCTION Function
//-------------------------------------
void TSLCDOutIns(word ins)
{
                addr_ptr = XPAR_LCD_CONTROL_0_BASEADDR + 12;
                *addr_ptr = ins;

}


//-----------------------------------------
//        Reset Function
```

```
//-----------------------------------------
void TSLCDRst()
	{
		setBitRST;
		clrBitRST;
		delay(100);
		setBitRST;
	}

//-----------------------------------------------------------------
//	Initial LCD Function
//-----------------------------------------------------------------
void TSLCDInit()
{
	delay(1000);
	TSLCDOutIns(0x00E5);
	TSLCDOutDat(0x8000);				//set the internal
vcore voltage
	TSLCDOutIns(TS_INS_START_OSC);
	TSLCDOutDat(0x0001);				//start oscillator

	delay(5000);

	TSLCDOutIns(TS_INS_DRIV_OUT_CTRL);
	TSLCDOutDat(0x0100);				//set SS, SM
	TSLCDOutIns(TS_INS_DRIV_WAV_CTRL);
	TSLCDOutDat(0x0700);				//set 1 line
inversion

	TSLCDOutIns(TS_INS_ENTRY_MOD);
	TSLCDOutDat(TS_VAL_ENTRY_MOD);			//set GRAM write
direction, BGR=0

	TSLCDOutIns(TS_INS_RESIZE_CTRL);
	TSLCDOutDat(0x0000);				//no resizing

	TSLCDOutIns(TS_INS_DISP_CTRL2);
	TSLCDOutDat(0x0202);				//front & back
porch periods = 2
	TSLCDOutIns(TS_INS_DISP_CTRL3);
	TSLCDOutDat(0x0000);
	TSLCDOutIns(TS_INS_DISP_CTRL4);
	TSLCDOutDat(0x0000);
	TSLCDOutIns(TS_INS_RGB_DISP_IF_CTRL1);
	TSLCDOutDat(0x0000);				//select system
interface
	TSLCDOutIns(TS_INS_FRM_MARKER_POS);
	TSLCDOutDat(0x0000);
	TSLCDOutIns(TS_INS_RGB_DISP_IF_CTRL2);
```

```
TSLCDOutDat(0x0000);

TSLCDOutIns(TS_INS_POW_CTRL1);
TSLCDOutDat(0x0000);
TSLCDOutIns(TS_INS_POW_CTRL2);
TSLCDOutDat(0x0000);
TSLCDOutIns(TS_INS_POW_CTRL3);
TSLCDOutDat(0x0000);
TSLCDOutIns(TS_INS_POW_CTRL4);
TSLCDOutDat(0x0000);
delay(20000);

TSLCDOutIns(TS_INS_POW_CTRL1);
TSLCDOutDat(0x17B0);
TSLCDOutIns(TS_INS_POW_CTRL2);
TSLCDOutDat(0x0137);
delay(5000);

TSLCDOutIns(TS_INS_POW_CTRL3);
TSLCDOutDat(0x013C);
delay(5000);

TSLCDOutIns(TS_INS_POW_CTRL4);
TSLCDOutDat(0x1400);
TSLCDOutIns(TS_INS_POW_CTRL7);
TSLCDOutDat(0x0007);
delay(50000);

TSLCDOutIns(TS_INS_GRAM_HOR_AD);
TSLCDOutDat(0x0000);
TSLCDOutIns(TS_INS_GRAM_VER_AD);
TSLCDOutDat(0x0000);

TSLCDOutIns(TS_INS_GAMMA_CTRL1);
TSLCDOutDat(0x0007);
TSLCDOutIns(TS_INS_GAMMA_CTRL2);
TSLCDOutDat(0x0504);
TSLCDOutIns(TS_INS_GAMMA_CTRL3);
TSLCDOutDat(0x0703);
TSLCDOutIns(TS_INS_GAMMA_CTRL4);
TSLCDOutDat(0x0002);
TSLCDOutIns(TS_INS_GAMMA_CTRL5);
TSLCDOutDat(0x0707);
TSLCDOutIns(TS_INS_GAMMA_CTRL6);
TSLCDOutDat(0x0406);
TSLCDOutIns(TS_INS_GAMMA_CTRL7);
TSLCDOutDat(0x0006);
TSLCDOutIns(TS_INS_GAMMA_CTRL8);
TSLCDOutDat(0x0404);
```

```
        TSLCDOutIns(TS_INS_GAMMA_CTRL9);
        TSLCDOutDat(0x0700);
        TSLCDOutIns(TS_INS_GAMMA_CTRL10);
        TSLCDOutDat(0x0A08);

        TSLCDOutIns(TS_INS_HOR_START_AD);
        TSLCDOutDat(0x0000);
        TSLCDOutIns(TS_INS_HOR_END_AD);
        TSLCDOutDat(0x00EF);
        TSLCDOutIns(TS_INS_VER_START_AD);
        TSLCDOutDat(0x0000);
        TSLCDOutIns(TS_INS_VER_END_AD);
        TSLCDOutDat(0x013F);
        TSLCDOutIns(TS_INS_GATE_SCAN_CTRL1);
        TSLCDOutDat(0x2700);
        TSLCDOutIns(TS_INS_GATE_SCAN_CTRL2);
        TSLCDOutDat(0x0001);
        TSLCDOutIns(TS_INS_GATE_SCAN_CTRL3);
        TSLCDOutDat(0x0000);

        TSLCDOutIns(TS_INS_PART_IMG1_DISP_POS);
        TSLCDOutDat(0x0000);
        TSLCDOutIns(TS_INS_PART_IMG1_START_AD);
        TSLCDOutDat(0x0000);
        TSLCDOutIns(TS_INS_PART_IMG1_END_AD);
        TSLCDOutDat(0x0000);
        TSLCDOutIns(TS_INS_PART_IMG2_DISP_POS);
        TSLCDOutDat(0x0000);
        TSLCDOutIns(TS_INS_PART_IMG2_START_AD);
        TSLCDOutDat(0x0000);
        TSLCDOutIns(TS_INS_PART_IMG2_END_AD);
        TSLCDOutDat(0x0000);

        TSLCDOutIns(TS_INS_PANEL_IF_CTRL1);
        TSLCDOutDat(0x0010);
        TSLCDOutIns(TS_INS_PANEL_IF_CTRL2);
        TSLCDOutDat(0x0000);
        TSLCDOutIns(TS_INS_PANEL_IF_CTRL3);
        TSLCDOutDat(0x0003);
        TSLCDOutIns(TS_INS_PANEL_IF_CTRL4);
        TSLCDOutDat(0x0110);
        TSLCDOutIns(TS_INS_PANEL_IF_CTRL5);
        TSLCDOutDat(0x0000);
        TSLCDOutIns(TS_INS_PANEL_IF_CTRL6);
        TSLCDOutDat(0x0000);

        TSLCDOutIns(TS_INS_DISP_CTRL1);
        TSLCDOutDat(0x0173);
}
```

```
//-----------------------------------------
//        Delay Function
//-----------------------------------------
void delay(unsigned int count)
{
unsigned int j, i;

//for(i=0; i<count; i++);
  //for(j=0; j<1; j++);
}

//-----------------------------------------
//        DelayMs Function
//-----------------------------------------
void delayMs(unsigned int count)
{
unsigned int j, i;

for(i=0; i<count; i++);
  for(j=0; j<1; j++);
}


//-----------------------------------------
//        FillRect Function
//-----------------------------------------
void FillRect(ts_pos_t sx,ts_pos_t ex,ts_pos_t sy,ts_pos_t ey,word color)
{
        word x,y;
        word i,j;
        if (sx < ts_margin_xl)
                sx = ts_margin_xl;
        if (ex > ts_margin_xr)
                ex = ts_margin_xr;
        if (sy < ts_margin_yu)
                sy = ts_margin_yu;
        if (ey > ts_margin_yl)
                ey = ts_margin_yl;

        TSLCDOutIns(TS_INS_START_ADX);
        TSLCDOutDat(sx);
        TSLCDOutIns(TS_INS_END_ADX);
        TSLCDOutDat(ex);
        TSLCDOutIns(TS_INS_GRAM_ADX);
        TSLCDOutDat(sx);
        x = ex - sx + 1;
```

```
#ifndef TS_ORN_PORTRAIT
	sy = TS_SIZE_Y - 1 - sy;	// mirror start y address
	ey = TS_SIZE_Y - 1 - ey;	// mirror end y address
	TSLCDOutIns(TS_INS_START_ADY);
	TSLCDOutDat(ey);
	TSLCDOutIns(TS_INS_END_ADY);
	TSLCDOutDat(sy);
	TSLCDOutIns(TS_INS_GRAM_ADY);
	TSLCDOutDat(sy);//fix from bug of v1_00
	y = sy - ey + 1;
#else
	TSLCDOutIns(TS_INS_START_ADY);
	TSLCDOutDat(sy);
	TSLCDOutIns(TS_INS_END_ADY);
	TSLCDOutDat(ey);
	TSLCDOutIns(TS_INS_GRAM_ADY);
	TSLCDOutDat(sy);
	y = ey - sy + 1;
#endif

	TSLCDOutIns(TS_INS_RW_GRAM);

		for (j=0; j<y; j++)
			for (i=0; i<x; i++)
			{
				TSLCDOutDat(color);
			}
}

//-----------------------------------------
//	SetFontType
//-----------------------------------------
void SetFontType(int font_no) //set text's color
{
	font_type = font_no;
	//xil_printf("font %d \r\n",font_type);
}

//-----------------------------------------
//	SetFontColor
//-----------------------------------------
void SetFontColor(word color) //set text's color
{
	font_color = color;
}

//-----------------------------------------
//	SetBackColor
//-----------------------------------------
```

```
void SetBackColor(word color) //set back color
{
	back_color = color;
}

//---------------------------------------------------------------------
//	PrintStr Function
//---------------------------------------------------------------------
void PrintStr(byte line,byte column,char *str)
{
	int i = 0;
	ts_pos_t posx,posy;
	posx = offsetx + column*FONT_WIDTH;
	posy = offsety + line*FONT_HEIGHT;

	while(str[i])
	{
		CharDisp(str[i],posx,posy);
		posx += FONT_WIDTH;
		i++;
	}
}

//---------------------------------------------------------------------
//	Show Pic Function
//---------------------------------------------------------------------
void ShowPic(ts_pos_t sx,ts_pos_t ex,ts_pos_t sy,ts_pos_t ey,const unsigned short
*pic)
{
	unsigned long k = 0;
	unsigned short color;
	unsigned int x,y;
	unsigned int i,j;
	if (sx < ts_margin_xl)
		sx = ts_margin_xl;
	if (ex > ts_margin_xr)
		ex = ts_margin_xr;
	if (sy < ts_margin_yu)
		sy = ts_margin_yu;
	if (ey > ts_margin_yl)
		ey = ts_margin_yl;

	TSLCDOutIns(TS_INS_START_ADX);
	TSLCDOutDat(sx);
	TSLCDOutIns(TS_INS_END_ADX);
	TSLCDOutDat(ex);
	TSLCDOutIns(TS_INS_GRAM_ADX);
	TSLCDOutDat(sx);
	x = ex - sx + 1;
```

```
#ifndef TS_ORN_PORTRAIT
	sy = TS_SIZE_Y - 1 - sy;	// mirror start y address
	ey = TS_SIZE_Y - 1 - ey;	// mirror end y address
	TSLCDOutIns(TS_INS_START_ADY);
	TSLCDOutDat(ey);
	TSLCDOutIns(TS_INS_END_ADY);
	TSLCDOutDat(sy);
	TSLCDOutIns(TS_INS_GRAM_ADY);
	TSLCDOutDat(sy);//fix from bug of v1_00
	y = sy - ey + 1;
#else
	TSLCDOutIns(TS_INS_START_ADY);
	TSLCDOutDat(sy);
	TSLCDOutIns(TS_INS_END_ADY);
	TSLCDOutDat(ey);
	TSLCDOutIns(TS_INS_GRAM_ADY);
	TSLCDOutDat(sy);
	y = ey - sy + 1;
#endif

	TSLCDOutIns(TS_INS_RW_GRAM);

		for (j=0; j<y; j++)
			for (i=0; i<x; i++)
			{
				TSLCDOutDat(pic[k]);
				k++;
			}
}

//-----------------------------------------
//	SetBackGroundColor Function
//-----------------------------------------
void SetBackGroundColor (word color) {
	FillRect(0,TS_SIZE_X-1,0,TS_SIZE_Y-1,color);
	FillCirc(120,160,200,color);
}

//-------------------------------------------------------------------
//	Show Pic Function From RAM
//-------------------------------------------------------------------
void ShowPicRAM(ts_pos_t sx,ts_pos_t ex,ts_pos_t sy,ts_pos_t ey,Xuint32
resAdres)
{
	unsigned long k = XPAR_MICRON_RAM_MEM0_BASEADDR+resAdres;
	//k = XIo_In16(XPAR_MICRON_RAM_MEM0_BASEADDR);
	unsigned short color;
	unsigned int x,y;
```

```
	unsigned int i,j;
	if (sx < ts_margin_xl)
		sx = ts_margin_xl;
	if (ex > ts_margin_xr)
		ex = ts_margin_xr;
	if (sy < ts_margin_yu)
		sy = ts_margin_yu;
	if (ey > ts_margin_yl)
		ey = ts_margin_yl;

	TSLCDOutIns(TS_INS_START_ADX);
	TSLCDOutDat(sx);
	TSLCDOutIns(TS_INS_END_ADX);
	TSLCDOutDat(ex);
	TSLCDOutIns(TS_INS_GRAM_ADX);
	TSLCDOutDat(sx);
	x = ex - sx + 1;

#ifndef TS_ORN_PORTRAIT
	sy = TS_SIZE_Y - 1 - sy;	// mirror start y address
	ey = TS_SIZE_Y - 1 - ey;	// mirror end y address
	TSLCDOutIns(TS_INS_START_ADY);
	TSLCDOutDat(ey);
	TSLCDOutIns(TS_INS_END_ADY);
	TSLCDOutDat(sy);
	TSLCDOutIns(TS_INS_GRAM_ADY);
	TSLCDOutDat(sy);//fix from bug of v1_00
	y = sy - ey + 1;
#else
	TSLCDOutIns(TS_INS_START_ADY);
	TSLCDOutDat(sy);
	TSLCDOutIns(TS_INS_END_ADY);
	TSLCDOutDat(ey);
	TSLCDOutIns(TS_INS_GRAM_ADY);
	TSLCDOutDat(sy);
	y = ey - sy + 1;
#endif

	TSLCDOutIns(TS_INS_RW_GRAM);

		for (j=0; j<y; j++)
			for (i=0; i<x; i++)
			{
				TSLCDOutDat(XIo_In16(k));
				k+=2;
			}
}

//-----------------------------------------------------------------
```

```
//      FillCirc
//--------------------------------------------------------------------
void FillCirc(ts_pos_t cx,ts_pos_t cy,ts_pos_t rad,word color) //draw a circle
{
#ifndef TS_ORN_PORTRAIT
        int sy_buf,ey_buf;
#endif
        int sx,sy,ex,ey;
        int i,j;
        word color_buf;
        word rad2 = rad*rad;
        sx = cx - rad;
        ex = cx + rad;
        sy = cy - rad;
        ey = cy + rad;

        if (sx < ts_margin_xl)
                sx = ts_margin_xl;
        if (ex > ts_margin_xr)
                ex = ts_margin_xr;
        if (sy < ts_margin_yu)
                sy = ts_margin_yu;
        if (ey > ts_margin_yl)
                ey = ts_margin_yl;

        TSLCDOutIns(TS_INS_START_ADX);
        TSLCDOutDat(sx);
        TSLCDOutIns(TS_INS_END_ADX);
        TSLCDOutDat(ex);
        TSLCDOutIns(TS_INS_GRAM_ADX);
        TSLCDOutDat(sx);

#ifndef TS_ORN_PORTRAIT
        sy_buf = TS_SIZE_Y - 1 - sy;        // mirror start y address
        ey_buf = TS_SIZE_Y - 1 - ey;        // mirror end y address
        TSLCDOutIns(TS_INS_START_ADY);
        TSLCDOutDat(ey_buf);
        TSLCDOutIns(TS_INS_END_ADY);
        TSLCDOutDat(sy_buf);
        TSLCDOutIns(TS_INS_GRAM_ADY);
        TSLCDOutDat(sy_buf);//fix from bug of v1_00
#else
        TSLCDOutIns(TS_INS_START_ADY);
        TSLCDOutDat(sy);
        TSLCDOutIns(TS_INS_END_ADY);
        TSLCDOutDat(ey);
        TSLCDOutIns(TS_INS_GRAM_ADY);
        TSLCDOutDat(sy);
#endif
```

```
	TSLCDOutIns(TS_INS_RW_GRAM);

		for (j=sy-cy; j<=ey-cy; j++)
			for (i=sx-cx; i<=ex-cx; i++)
			{
				if ((i)*(i) + (j)*(j) < rad2)
				{
					TSLCDOutDat(color);
				}
				else
				{
					TSLCDOutDat(back_color);
				}
			}
}


//---------------------------------------------------------------------
//	CharDisp
//---------------------------------------------------------------------
void CharDisp(char charactor,ts_pos_t sx,ts_pos_t sy) //low level function to print a
character on LCD
{
	word x,y;
	byte i,j;
	ts_pos_t ex,ey;
	word c;

	ex = sx + FONT_WIDTH - 1;
	ey = sy + FONT_HEIGHT - 1;

	buf_store(charactor - 0x20);

	TSLCDOutIns(TS_INS_START_ADX);
	TSLCDOutDat(sx);
	TSLCDOutIns(TS_INS_END_ADX);
	TSLCDOutDat(ex);
	TSLCDOutIns(TS_INS_GRAM_ADX);
	TSLCDOutDat(sx);
	x = ex - sx + 1;

#ifndef TS_ORN_PORTRAIT
	sy = TS_SIZE_Y - 1 - sy;	// mirror start y address
	ey = TS_SIZE_Y - 1 - ey;	// mirror end y address
	TSLCDOutIns(TS_INS_START_ADY);
	TSLCDOutDat(ey);
	TSLCDOutIns(TS_INS_END_ADY);
	TSLCDOutDat(sy);
```

```
        TSLCDOutIns(TS_INS_GRAM_ADY);
        TSLCDOutDat(sy);//fix from bug of v1_00
        y = sy - ey + 1;
#else
        TSLCDOutIns(TS_INS_START_ADY);
        TSLCDOutDat(sy);
        TSLCDOutIns(TS_INS_END_ADY);
        TSLCDOutDat(ey);
        TSLCDOutIns(TS_INS_GRAM_ADY);
        TSLCDOutDat(sy);
        y = ey - sy + 1;
#endif

        TSLCDOutIns(TS_INS_RW_GRAM);


                for (j=0; j<y; j++)
                        for (i=0; i<x; i++)
                        {
                                if (buf_read(i,j))
                                {
                                        TSLCDOutDat(font_color);
                                }
                                else
                                {
                                        TSLCDOutDat(back_color);
                                }
                        }
}


//-------------------------------------------
//      TSLCDSetMargins Function
//-------------------------------------------
void TSLCDSetMargins(ts_pos_t xl,ts_pos_t xr,ts_pos_t yu,ts_pos_t yl) //set
margins for FillRect,FillCirc
{
        ts_margin_xl = xl;
        ts_margin_xr = xr;
        ts_margin_yu = yu;
        ts_margin_yl = yl;
}

//-------------------------------------------
//      TSLCDSetMarginsDefault Function
//-------------------------------------------
void TSLCDSetMarginsDefault(void) //Reset margins to default value
{
        ts_margin_xl = 0;
```

```
	ts_margin_xr = TS_SIZE_X - 1;
	ts_margin_yu = 0;
	ts_margin_yl = TS_SIZE_Y - 1;
}

//-------------------------------------------
//	buf_read Function
//-------------------------------------------
byte buf_read(byte column,byte row)
{
	byte read_pixel;

	if (row < 8)
	{
		read_pixel = (char_buf[column][0] >> (7-row)) & 0x01;
	}
	else
	{
		row = row - 8;
		read_pixel = (char_buf[column][1] >> (7-row)) & 0x01;
	}
	return (read_pixel);
}

//-------------------------------------------
//	buf_store Function
//-------------------------------------------
void buf_store(byte charactor)
{
	byte i,j;
	int char_p,fontSize,fontBitWidth,fontByteHeight;
	//fontSize=16;
	//xil_printf("font turu-- %d ",font_type);
		if (font_type==0)	{
			fontBitWidth=11;
			fontByteHeight=2;
		}
		else if (font_type==1){
			fontBitWidth=8;
			fontByteHeight=2;
		}
		else if (font_type==2){
			fontBitWidth=18;
			fontByteHeight=2;
		}
		fontSize =fontBitWidth * fontByteHeight;
		char_p = charactor*fontSize; //FONT SIZE
		//xil_printf("char p-- %d ",char_p);
	for (i=0; i<fontBitWidth; i++)
```

```
                for (j=0; j<fontByteHeight; j++)
                {
                        if (font_type==0)       {
                                char_buf[i][j] = font_vrinda11x16[char_p];
                        }
                        else if (font_type==1){
                                char_buf[i][j] = font_courier[char_p];
                        }
                        else if (font_type==2){
                                char_buf[i][j] = font_vrinda18x16[char_p];
                        }
                        char_p++;
                }
}
```

ÖZGEÇMİŞ

1984 yılında Afyonkarahisar'da doğdu. İlk, orta ve lise öğrenimini Afyonkarahisar'da tamamladı. 2005 yılında girdiği Kocaeli Üniversitesi Teknik Eğitim Fakültesi Elektronik ve Bilgisayar Eğitimi Bölümü Bilgisayar Öğretmenliği dalından 2009'da Bilgisayar Öğretmeni olarak mezun oldu. 2009 yılında Kocaeli Üniversitesi Fen Bilimleri Enstitüsü, Elektronik ve Bilgisayar Eğitimi Anabilim Dalı'nda Yüksek Lisans öğrenimine başladı. Halen yüksek lisans öğrenimine devam etmektedir.