T.C.
BOLU ABANT İZZET BAYSAL ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ


# FPGA İLE ZAMAN-SAYISAL DÖNÜŞTÜRÜCÜ TASARIMI

YÜKSEK LİSANS TEZİ

**MERT SONGEL**

**BOLU, TEMMUZ - 2019**

T.C.

BOLU ABANT İZZET BAYSAL ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI


# FPGA İLE ZAMAN-SAYISAL DÖNÜŞTÜRÜCÜ TASARIMI

YÜKSEK LİSANS TEZİ

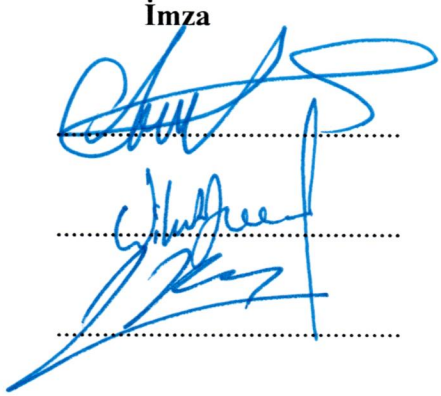
MERT SONGEL

BOLU,TEMMUZ - 2019

# KABUL VE ONAY SAYFASI

**MERT SONGEL** tarafından hazırlanan "**FPGA İLE ZAMAN - SAYISAL DÖNÜŞTÜRCÜ TASARIMI**" adlı tez çalışması Elektrik Elektronik Mühendisliği Anabilim Dalı'nda 29.07.2019 tarihinde **Bolu Abant İzzet Baysal Üniversitesi Fen Bilimleri Enstitüsü** Yüksek Lisans Tezi olarak kabul edilmiştir.

| **Jüri Üyeleri** | **İmza** |
|---|---|
| Danışman<br>**Dr. Öğr. Üyesi Oktay AYTAR** | ..... |
| Üye<br>**Dr. Öğr. Üyesi Nihat DALDAL** | ..... |
| Üye<br>**Dr. Öğr. Üyesi Selman KULAÇ** | ..... |



**Prof. Dr. Ömer ÖZYURT** .....

**Fen Bilimleri Enstitüsü Müdürü** V.

**Eşime ve çocuğuma,**

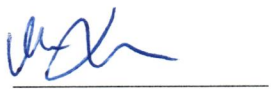
## ETİK BEYAN

Abant İzzet Baysal Üniversitesi, Fen Bilimleri Enstitüsü Tez Yazım Kurallarına uygun olarak hazırladığım bu tez çalışmasında;

- Tez içinde sunduğum verileri, bilgileri ve dokümanları akademik ve etik kurallar çerçevesinde elde ettiğimi,
- Tüm bilgi, belge, değerlendirme ve sonuçları bilimsel etik ve ahlak kurallarına uygun olarak sunduğumu,
- Tez çalışmasında yararlandığım eserlerin tümüne uygun atıfta bulunarak kaynak gösterdiğimi,
- Kullanılan verilerde herhangi bir değişiklik yapmadığımı,
- Bu tezde sunduğum çalışmanın özgün olduğunu,

bildirir, aksi bir durumda aleyhime doğabilecek tüm hak kayıplarını kabullendiğimi beyan ederim.

**Mert SONGEL**



# ÖZET


**FPGA İLE ZAMAN-SAYISAL DÖNÜŞTÜRÜCÜ TASARIMI**
**YÜKSEK LİSANS TEZİ**
**MERT SONGEL**
**BOLUABANT İZZET BAYSAL ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ**
**ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI**
**(TEZ DANIŞMANI:DR. ÖĞR. ÜYESİ, OKTAY AYTAR)**

**BOLU, TEMMUZ - 2019**

Zaman-Sayısal Dönüştürücü, başlangıç ve bitiş olarak adlandırılan iki darbe işareti arasında geçen süreyi sayısal koda dönüştürmektedir. Özellikle medikal tarama cihazları, x-rayler, radarlar, sensörler ve nükleer fizik çalışmalarında çok hassas zaman aralığına ihtiyaç duyulmaktadır. Bu yüzden, bu uygulamaların kullanıldığı sistemlerin en önemli bloklarından birisidir. Zaman-Sayısal Dönüştürücü; Tümüyle Paralel (Flash),Vernier gecikme hatlı, Vernier halka gecikmeli ve Sayıcı Tabanlı gibi farklı yöntemler ile tasarlanabilir. Yapılan bu çalışmada sayıcı tabanlı yöntem kullanılmıştır. Buna bağlı olarak da sayısal devre tasarımında sağladığı avantajlar, esneklikler, paralel çalışma ve güç kullanımındaki verimlilik nedeniyle de bu çalışmada FPGA tercih edilmiştir. FPGA'in programlanması için ISE Design Suite 14.7 programında VHDL dili kullanılmıştır.

Yapılan sistem tasarımında, toplam dört adet sinyal girişi bulunmaktadır. Başlama, durdurma, sıfırlama ve sistemin darbe işareti olarak isimlendirilen bu girişlerin yanı sıra, sistemin yeniden başlatılması için de ayrıca bir sıfırlama butonu bulunmaktadır. Başlangıç ve durdurma işaretleri kullanıcı tarafından girilen bu tasarım da 20MHz, 40MHz, 50MHz ve 66MHz olmak üzere dört farklı darbe üreteci ile zaman-sayısal dönüştürücünün benzetim sonuçları alınmıştır. Aynı zamanda Spartan 3E başlangıç kartına aktarılarak donanımsal gerçekleştirilmesi de yapılmıştır. Yapılan benzetim ve donanımsal gerçekleme sonucunda 66 MHz'lik darbe üreteci kullanıldığında sisteme uygulanabilecek en az başlangıç süresinin 15 ns, bitiş süresinin ise en fazla 65 sn olduğu gözlemlenmiştir.



**ANAHTAR KELİMELER: Zaman-Sayısal Dönüştürücü, FPGA, Spartan 3E başlangıç kartı, HDL, VHDL.**

# ABSTRACT


**TIME TO DIGITAL CONVERTER DESIGN WITH FPGA**
**PHD THESIS**
**MERT SONGEL**
**BOLU ABANT IZZET BAYSAL UNIVERSITYGRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES**
**DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING**
**(SUPERVISOR:ASSIST.PROF.DR., OKTAY AYTAR)**

**BOLU, JULY 2019**

The Time to Digital Converter (TDC) converts the passing time between two pulse signals called as "start" and "stop" into the numerical code. Particulary, medical scanning equipments, x-rays, radars, sensors and nuclear physics are requied highly sensitive time period. therefore, it is one of the significant component in these kinds of systems. TDC can be designed with different methods such as Flash, Vernier's delay line, Vernier's ring delay line and Counter based. Counter based was used in this study. Correspondingly, FPGA was preffered in this study due to its advantages and flexibilities at digital circuits design and the ability of paralel working and effectiveness at the usage of power. FPGA was programmed with ISE Design Suite 14.7 by using VHDL.

In the system design, there are a total of four signal inputs. In addition to the so-called start, stop, reset and pulse signal of the system, there is also a reset button for restarting the system. In this design, where start and stop signals are entered by the user, the simulation results of four different pulse generators, 20MHz, 40MHz, 50MHz and 66MHz, and the time-digital converter are obtained. At the same time, it was transferred to Spartan 3E starter card and its hardware implementation was realized. As a result of the simulation and hardware implementation realization, when the pulse generator of 66 MHz was used, it was observed that the minimum start time was 15 ns and the maximum end time was 65 seconds.



**KEYWORDS: Time to Digital Converter, FPGA, Spartan 3E starter board, HDL, VHDL.**

# İÇİNDEKİLER

**Sayfa**

# ŞEKİL LİSTESİ

**Sayfa**

# KISALTMA VE SEMBOLLER LİSTESİ

| | |
|---|---|
| **ZSD** | **:** Zaman / Sayısal Dönüştürücü. |
| **IEEE** | **:**The Institute of Electrical and Electronics Engineers (Elektrik-Elektronik Mühendisleri Enstitüsü). |
| **VHDL** | **:** Very High –Speed Integrated Circuit Hardware Description Language ( Yüksek Hızlı Donanım Tanımlama Dili). |
| **HDL** | **:** Hardware Description Language (Donanım Tanımlama Dili). |
| **sn** | **:** Saniye. |
| **ns** | **:** Nano Saniye. |
| **ns** | **:** Nano Metre. |
| **MHz** | **:** Mega Hertz. |
| **THz** | **:** Tera Hertz. |
| **LCD** | **:** Liquid Crystal Display (Sıvı Kristal Ekran). |
| **FPGA** | **:** Field Programmable Gate Array (Alanda Programlanabilir Kapı Dizileri). |
| **CPLD** | **:**Complex Programmable Logic Device (Karmaşık Programlanabilir Mantık Cihazı). |
| **3DIC** | **:**3 Dimensional integrated circuit ( 3 boyutlu entegre devre). |
| **SoC** | **:**System on Chip (Çip Üzerine Sistem). |
| **USB** | **:**Universal Serial Bus (Evrensel Seri Veri yolu). |
| **JTAG** | **:**Joint Test Action Group (ortak test eylem grubu). |
| **DIP** | **:**dual in-line package (çift hat içi paket). |
| **SHA** | **:**Safe hash algorithm (Güvenli Karma Algoritma). |
| **DAC** | **:**Digital to Anolog Converter (Sayısal Analog dönüştürücü). |
| **ADC** | **:**Analog to Digital Converter (Analog Sayısal Dönüştürücü). |
| **SMA** | **:**Sub Miniature version A(Alt minyatür versiyonu A). |
| **CLB** | **:**Configurable Logic Block (Yapılandırılabilir Mantık Bloğu). |
| **LUT** | **:**LookUp Table (arama tablosu). |
| **I / O** | **:**In / Out (Giriş / Çıkış). |
| **DDR** | **:**Double Data Rate ( Çift Veri Hızı). |
| **SPI** | **:**Serial Peripheral Interface (Seri Çevre Ara yüzü). |
| **RISC** | **:**Reduced instruction set computing (Azaltılmış komut seti |

hesaplama).

**PCI** :Peripheral Component Interconnect (Çevre Birim Bileşen Bağlantısı).

**Bit** :Veri uzunluk birimi.

**MBit** :Mega bit.

**MByte** :Mega Byte. (1 Byte = 8 Bit).

**ASIC** :Application Specific Integrated Circuit(Uygulamaya Özel Tümleşik Devre).

**SRAM** :Synchronous Random Access Memory (Senkron Rasgele Erişim Belleği).

**SDRAM** :Synchronous Dynamic Random Access Memory (Senkron Dinamik Rasgele Erişim Belleği).

**ROM** :Read Only Memory ( Sadece Okunabilir bellek).

**PROM** :Programmable ROM ( programramlanabilir ROM)

**EPROM** :Erasable PROM ( silinebilir PROM).

**EEPROM** :Electrically EPROM ( Elektriksel olarak EPROM).

**NOR** :Veya Kapısının terslenmişi.

**UART** :Universal asynchronous receiver-transmitter (Evrensel asenkron alıcı-verici).

**VGA** :Video Graphics Array (Video grafik dizisi).

**RS-232** :Recommended Standard 232 (Tavsiye Edilen Standart 232).

**DTE** :Data terminal equipment (veri terminal ekipmanı).

**DCE** :Data circuit-terminating equipment (veri taşıma ekipmanı).

**SHA** :Secure Hash Algorithms (Güvenli Karma Algoritma).

# TEŞEKKÜR

Öncelikle bana her konuda elinden gelen desteği esirgemeyen danışman hocam Oktay AYTAR'a, literatür taramasında yabancı dilde yetersiz kaldığım kısımlarda yardım eden Enes ASLAN arkadaşıma, beni bu günlere kadar getiren ellerinden gelenin fazlasını yapmaya çalışan aileme ve bu tezi yazarken dünyaya gözlerini geçtiğimiz sene açan evladımın her türlü sorunuyla tek başına uğraşmak zorunda kalan dünyalar güzeli eşime canı gönülden teşekkürlerimi sunuyorum. Ayrıca fırsat buldukça beni hiç tanımadan yardım etmeye çalışan Mehmet Ali ÇAVUŞLU'ya da teşekkürlerimi borç bilirim.

# 1. GİRİŞ

Başlangıç ve bitiş zamanı arasında geçen süre ölçümünün önemli olduğu birçok uygulamalı sistem vardır. Özellikle zamana bağlı sinyal işleme ve yüksek enerji fiziği uygulamaları, medikal görüntüleme sistemleri, sayısal osiloskop ve lojik analizör cihazlarının kullanıldığı uygulamalarda Zaman/Sayısal Dönüştürücüler (ZSD) büyük önem taşımaktadır (Wang vd., 2017; Balla vd., 2012; Zhang vd., 2018; Henzler, 2010; Büchele vd., 2016).

ZSD tasarımı yapılırken kullanım alanı, tasarım teknolojisi, gücü, hızı ve kapladığı alan dikkate alınır. Aynı zamanda ZSD tasarımı için literatürde farklı metotlar denenmektedir. Bu metotlardan bazıları Vernier gecikme hatlı ZSD, Vernier halka gecikmeli ZSD, Tümüyle paralel ZSD, Sayıcı Tabanlı ZSD'dir (Van den Broek, 2012).

Vernier'in gecikme hatlı ZSD'si, literatürde sık rastlanan tasarım metotlardan biridir. Bu metodun tercih edilmesinin genel sebebi tasarlanan ZSD'nin yüksek çözünürlüklü olmasıdır (Jovanovi´c ve Stoj^cev, 2009; Dudek vd., 2000). Fakat Vernier gecikme hatlı ZSD tasarımında çözünürlük sayısı arttıkça hata faktörleri anlık darbe ölçümlerinde artmaktadır (Dudek vd., 2000).

Vernier'in halka gecikmeli ZSD'si literatür aramalarında sık rastlanan ikinci tasarım metodudur. Bu metot Vernier'in gecikme hatlı ZSD'nin son bitindeki gecikme bloğunun, ilk bitindeki gecikme bloğuna geri besleme yapılarak elde edilmektedir. Bu ZSD daha iyi çözünürlük, daha uzun zaman aralığı ölçümü, daha küçük boyutta üretim olanağı ve daha küçük güç tüketimi sağlamaktadır (Yu vd., 2010).

Tümüyle paralel ZSD, en basit ZSD yapılarından birisidir. Analog-sayısal dönüştürücüler de kullanılan ve referans gerilimlerinin elde edildiği dirençler yerine, gecikmeyi sağlayacak olan gecikme devreleri yerleştirilmiştir. Gecikme devresi olarak da tek çıkışlı veya farksal CMOS inverter devreleri, sayısal tampon devreleri kullanılabilir. Bu yapının en önemli dezavantajlarından birisi sistemin

çözünürlüğünün gecikme elemanının yayılım gecikmesi ile sınırlı olmasıdır. Aynı zamanda sistemin örnekleme oranı gecikme hattının toplam gecikmesi ile sınırlıdır (Van den Broek, 2012).

Sayıcı tabanlı ZSD giriş işaretlerini alıp bir sayıcıya göndererek burada sayma işlemini gerçekleştirmektedir. Durdur işareti geldiğinde çıkan sonuç kod çözücüye gönderilip çıkış işareti sağlanmaktadır. Bu ZSD'nin çözünürlüğü sistemin darbe işaretine, ölçebileceği zaman aralığı sayıcının kaç bitlik olduğuna bağlıdır. Maksimum hata oranı sistem darbe işaretinin bir periyodu kadardır (Keränen, 2016). Bit sayısını tasarımcı ne kadar arttırırsa arttırsın sonuç kararlı olmaktadır. Bu durum daha hassas hesaplama yapan uygulamalar için büyük bir avantaj sağlamaktadır.

ASIC ve FPGA teknolojisi kullanarak tasarlan ZSD nanosaniye (ns) altında ölçüm yapmak için yaygın olarak kullanılmaktadırlar. Tasarım boyutu küçüldükçe maliyet olarak FPGA teknolojisi ASIC teknolojisine göre avantajlı hale gelmektedir (Aloisio vd., 2008, 2009).

FPGA tasarımında kullanılan yonga, VHDL ya da Verilog dili kullanılarak ihtiyaca göre programlanabilir. Bu özelliğinden dolayı kullanımı oldukça basittir ve tasarım kolay bir şekilde değiştirilmektedir. Örneğin lojik VE kapısı olarak tasarlanan bir FPGA yongası yeniden programlanarak XOR kapısı olarak kullanılabilir. FPGA yongası ihtiyaca göre daimi olarak yeniden programlanma özelliğine sahiptir.

Literatürde farklı sistemlerde mikroişlemci, mikrodenetleyici, elektronik devre elemanları ve elektrik şalt malzemeleri kullanılarak çözüme gidilmektedir. Fakat bu sistemler de tasarlanan proje büyüdükçe, maliyet ve tasarım zorluğu artmaktadır. Dolayısıyla tasarımcılar maliyeti azaltan ve tasarımı kolaylaştıran sistemlere yönelmişlerdir. Tasarımını değiştirmek veya geliştirmek isteyen tasarımcı FPGA teknolojisi kullanarak sürekli devre elemanı değiştirmediğinden maliyet ve zaman açısından avantaj kazanmaktadır. Günümüzde büyük sanayi sistemlerinde maliyet ve işçilik göz önünde bulundurulduğunda FPGA yongası kullanılarak yapılan sistemler tercih edilmektedir.

FPGA teknolojisini rahat bir şekilde kullanılması ve test edilmesi için bazı firmalar kendilerine ait test veya deney kartları geliştirmişlerdir. Bu firmalardan en büyük iki isim Xilinx ve Altera'dır. Xilinx firmasının kendine ait deney ve geliştirme kartları bulunmaktadır. Bunlar Artix ailesi, Virtex ailesi, Zynq ailesi, Kintex ailesi ve Spartan ailesidir. Her bir geliştirme kartı ailesinin ortak ve birbirlerinden farklı özellikleri vardır. Tasarımcılar en uygun kart seçimini Xilinx firmasının resmi web sitesinden yapabilmektedirler.

Dadouche vd. (2015) "Design Methodology of TDC on Low Cost FPGA Targets" makalesinde, düşük maliyetli FPGA yongalarında yüksek çözünürlük sağlayan genel bir ZSD tasarım ve uygulama metodolojisi önermişlerdir. Bu metodolojinin hem ağdaki sinyal gecikmelerinde hem de FPGA yongalarının kendi gecikme kapılarında daha iyi sonuçlara olanak verdiğini belirtmişlerdir. Elde ettiklerin sonuçların, sadece yüksek performans gerektiren alanlar için değil, aynı zamanda çok sayıda hızlı uygulama alanında da çok umut verici olduğunu ifade etmişlerdir.

Sui vd. (2018) "A 2.3-ps RMS Resolution Time-to-Digital Converter Implemented in a Low-Cost Cyclone V FPGA" makalesinde, FPGA teknolojisinde gecikme hattı metodu yerine NUMP (nonuniform multiphase) metodu kullanarak düşük maliyetli bir ZSD tasarımı önermişlerdir. NUMP metodu, sistem darbe işaretine farklı faz açıları uygulanarak sistemde birden fazla yeni darbe işaretinin elde edilmesi ve bu işaretlerin gecikme hattındaki gecikme bloklarının yerine kullanılmasıdır. NUMP metodu kullanılarak oluşturulan ZSD'nin yüksek çözünürlüklü, yüksek performanslı ve düşük maliyetli olduğunu belirtmişlerdir. Ancak bu tasarım metodunu çok kanallı ZSD uygulamaları için kullanılabileceğini ifade etmişlerdir.

Won ve Lee (2016) "Time-to-Digital Converter Using a Tuned-Delay Line Evaluated in 28-, 40-, and 45-nm FPGAs" makalesinde FPGA tabanlı ayarlanabilir gecikme hatlı ZSD'ler için bin-width ayarlama yöntemi önermişlerdir. Bu yöntemi 3 farklı FPGA geliştirme kartında denemişlerdir. Bunlar Kintex-7, Virtex-6 ve Spartan-6'dır. Amaçlarının eşsiz bir heterojen ZSD yapısı geliştirmek olduğunu belirtmişlerdir. Üç farklı FPGA geliştirme kartında da

bin-width ile ayarlanan ZSD'nin ölçüm belirsizliğini ve RMS nicemleme hatasını azalttığını, doğrusallığı ise arttırdığını ifade etmişlerdir.

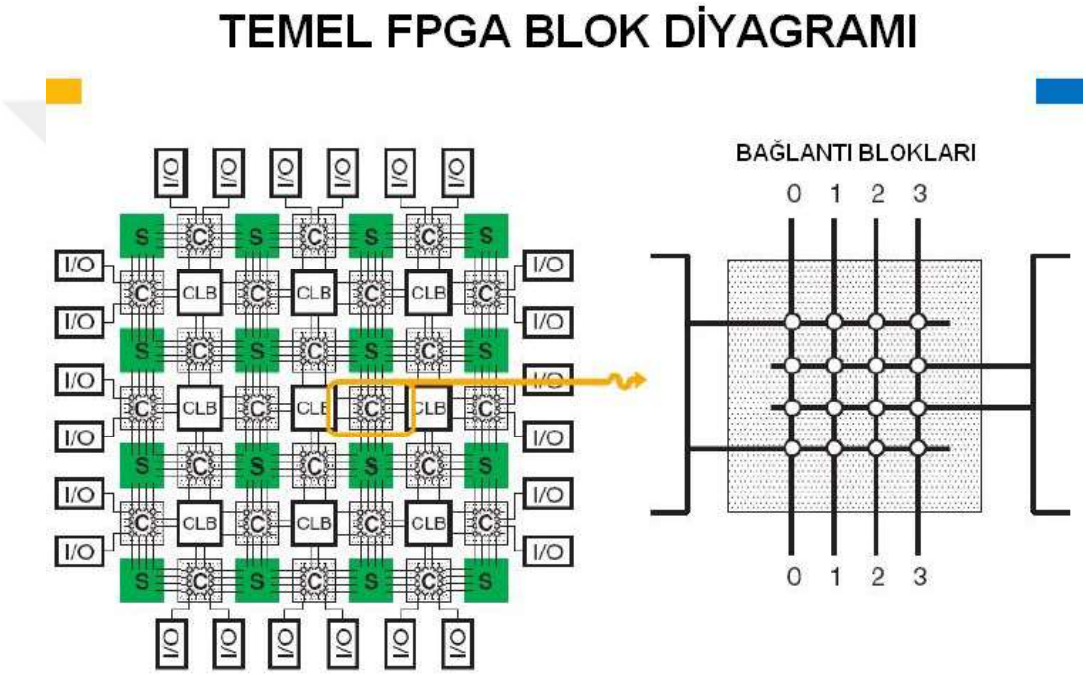
Soni vd. (2017) "Comparative Study of Delay Line Based Time to Digital Converter Using FPGA" makalesinde FPGA teknolojisi kullanarak birden fazla ZSD tasarım metodunu simülasyon ortamında test edip karşılaştırmışlardır. Elde ettikleri bu sonuçlara göre tasarım karmaşıklığı, çözünürlük ve teknoloji göz önüne alındığında Vernier'in halka gecikmeli ZSD'sinin daha uygun olduğunu belirtmişlerdir.

Mahalaxmi vd. (2016) "Time to Digital Converter (TDC) Implementation on Spartan 3e FPGA Using VHDL for TMTC Subsystems" makalesinde, uzay sistemi (TMTC) ve alt sistemleri arasındaki uydu iletişimi için VHDL kodu kullanarak ZSD uygulaması önermişlerdir. Yapılan bu tasarımda Spartan 3E başlangıç kartı kullanıldığı belirtilmiştir. Hassaslık kaybı ihmal edildiğinde, uygulanan test sinyallerinin düşme ve yükselme süresinin dikkate alınmamasından kaynaklanan küçük bir dezavantaj dışında sonuçların doğru olduğunu gözlemlemişlerdir. Daha hassas doğruluk için sinyallerin yükselme ve düşme süresini de içerecek şekilde bir çalışma yapıldığında yüksek hassasiyetli zaman ölçümü gerektiren uydu sistemi uygulamalarında osiloskop ihtiyacını ortadan kaldıracağını belirtmişlerdir.

Yapılan bu tez çalışmasının ikinci bölümünde FPGA, mimarisi ve donanım tanımlama dilleri, üçüncü bölümde ZSD tasarımında kullanılan sayıcılar, dördüncü bölümünde ise ZSD'ler genel olarak anlatılmıştır. Beşinci bölümde çalışmada kullanılan Spartan 3E başlangıç kartı tanıtılmış olup, altıncı bölümde ise tasarımı yapılan ZSD'nin simülasyon ortamında ve Spartan 3E başlangıç kartında elde edilen sonuçları gösterilmiştir.

## 2. FPGA

Bir FPGA yongası, programlanabilir mantık blokları ve bu bloklar arasındaki ara bağlantılardan oluşur. FPGA'lar oldukça geniş uygulama alanında kullanılan sayısal tümleşik devrelerdir. Şekil 1'de FPGA sisteminin temel blok diyagramı verilmiştir.




**Şekil 1:**Temel FPGA blok diyagramı (Mutukuda, 2009, 14 Nisan 2019).

FPGA teknolojisinin tasarım dili HDL'dir. Yaygın olarak kullanılan iki farklı HDL dili vardır. Bunlar Verilog ve VHDL'dir. Verilog ve VHDL dilinin birbirleri üzerinde hiçbir üstünlüğü yoktur (FPGA, 14 Nisan 2019). Tasarımcılar genellikle hangi dili öğrenmişlerse o dil ile devam etmektedirler. İki farklı tasarım diliyle de aynı işlemler yapılmaktadır.

## 2.1 FPGA Mimarisi

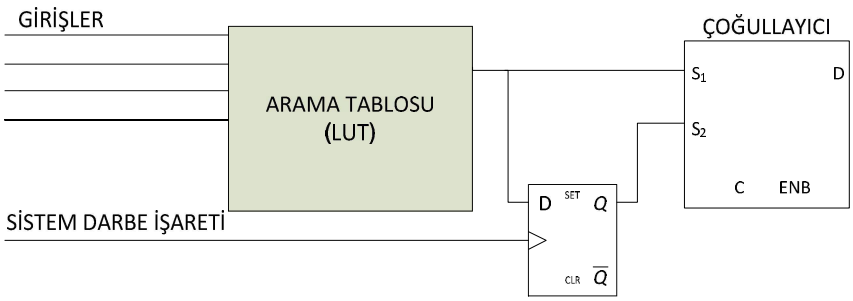
FPGA'lar temel olarak üç bloktan oluşur. Şekil 1'de de görüldüğü üzere bunlar yapılandırılabilir mantık blokları(CLB), giriş/çıkış blokları (I/O) ve bağlantı bloklarıdır. Bu blokların dışında aritmetik işlem blokları ve bellek blokları gibi özel bloklar içeren FPGA yapıları da mevcuttur (FPGA, 14 Nisan 2019).

Mantık blokları, FPGA içerisinde basit fonksiyonların gerçekleştirildiği yapılardır. FPGA içerisinde çarpma işlemini paralel olarak yürütebilen dijital sinyal işleme blokları ile sistem darbe işaretine bağlı olarak değişik frekans ve fazlarda çoklu darbe işaretleri üretebilen faz kilitli döngü modülleri de bulunmaktadır. Bu modüller tasarlanan devrede ihtiyaç duyulan darbe üreteci sayısını ve kullanılan darbe işareti pim sayısını azaltmaktadırlar.

### 2.1.1 Mantık Blokları

FPGA'nın ana yapısını mantık blokları oluşturmaktadır. Mantık bloğu; 1 adet arama tablosu (LUT), 1 adet D Flip-Flop'u ve 1 adet 2*1'lik çoğullayıcı'dan (multiplexer) oluşmaktadır. Şekil 2'de mantık bloğunun temel yapısı gösterilmiştir (FPGA, 14 Nisan 2019).

Arama tabloları bir mantık işlemini yerine getiren küçük belleklerdir. Binlerce mantık bloğunun birleşimi sonucunda kompleks ve büyük programlar oluşturmaktadır.




**Şekil 2:** Mantık bloğu temel yapısı.

### 2.1.2 Giriş/Çıkış Blokları

FPGA yongası içerisinde bulunan mantık yapıları ile FPGA pimleri, giriş/çıkış blokları ile bağlantı oluştururlar. Giriş/çıkış pimleri tasarlanan yapıya göre giriş, çıkış veya hem giriş hem çıkış olarak tanımlanabilirler (Sarıtaş ve Karataş, 2015).

### 2.1.3 Alıcı/Verici Blokları

Bu bloklar FPGA ile bağlı olduğu sistem veya sistemler arasında yapılan yüksek hızlı veri alışverişi sağlamaktadır. İçerisinde seri-paralel ya da paralel-seri dönüştürücüler ile darbe işareti blokları bulunmaktadır (Sarıtaş ve Karataş, 2015).

### 2.1.4 Programlanabilir Ara Bağlantılar

FPGA yongasındaki mantık hücrelerinin programlanabilir ara bağlantıları, veri yolları ve programlanma yeteneği olan anahtarlar yoluyla gerçekleşmektedir. FPGA yongası, donanım tanımlama diliyle oluşturulan program vasıtasıyla anahtarlama işlemi yapabilmektedir (Sarıtaş ve Karataş, 2015).

## 2.2 FPGA Programlama

FPGA'de iki adet çalışma modu vardır, bunlar konfigürasyon ve kullanıcı modudur. Enerji verildiği zaman FPGA yongası, konfigürasyon moduna geçer. Konfigürasyon modunda FPGA'in bütün çıkışları pasif olup, programlanması yapılır ve pimlerin durumları belirtilir. Yükleme bittiğinde FPGA yongası kullanıcı moduna, pasif olan pimler ise aktif duruma geçer (Sarıtaş ve Karataş, 2015).

FPGA iç bağlantılarının yapılması ancak programlanması ile mümkündür. Programlanabilir anahtarlama teknolojisi üreticiden üreticiye değişse de, bütün

üreticilerin kullandığı yöntem aslında aynıdır. Bu yöntem anahtarların açık ya da kapalı konumlarının belirlenmesidir (FPGA, 14 Nisan 2019).

Programlanabilir ara bağlantıların yapılması için farklı programlama teknolojileri bulunmaktadır. Bunlardan bazıları aşağıda verilmiştir;

- SRAM tabanlı,
- SRAM/Tümüyle paralel tabanlı,
- Sigortalamasız (Anti-fuse) tabanlı,
- EPROM/EEPROM tabanlı,
- Tümüyle paralel tabanlı (Sarıtaş ve Karataş, 2015).

Literatürde FPGA yongası programlamada sık rastlanan yöntemler SRAM, sigortalamasız ve EEPROM yöntemleridir. SRAM'de geçiş transistorlarının ya da mantık kapılarının kontrol edilmesi için küçük statik bellekler kullanılır. SRAM/Tümüyle paralel'de dâhili tümüyle paralel bellekler bulunmaktadır. Bu durumdan dolayı yapılandırma mekanizmasına ihtiyaç duymamaktadır. Sigortalamasız yöntem bir kez programlanabilir özelliğe sahiptir. Güç tüketimleri ve iletim gecikmeleri düşüktür. EPROM/EEPROM silinebilme ve yeniden programlanabilme özelliğine sahiptir. EPROM yöntemi belli süre ultraviyole ışınlarına maruz bırakılarak silinmektedir. EEPROM yöntemi ise üzerine yüksek gerilim uygulanarak silinebilir. Tümüyle paralel tabanlı yapılandırma verisi ayrı bir bellekte saklanır. Güç tüketimleri ve elektromanyetik alandan etkilenme oranları düşüktür (Sarıtaş ve Karataş, 2015).

## 2.3 Verilog-HDL Donanım Tanımlama Dili

Verilog-HDL, donanım tanımlama dillerinden birisidir. Donanım tanımlama dili, sayısal sistemleri tanımlamak için kullanılan dil demektir. Örnek verilecek olursa, ağ anahtarları, mikroişlemci, bellek veya flip-flop gibi. Tasarımcı, farklı seviyelerdeki herhangi bir donanımı tanımlayabilmek için HDL kullanabilir (Docplayer, 25 Nisan 2019).

### 2.3.1 Donanım Tanımlama Biçimleri

Çoğu donanım tanımlama dilinde olduğu gibi Verilog-HDL dilinde de aşağıdan yukarıya veya yukarıdan aşağıya metodoloji kullanılabilir (Docplayer, 25 Nisan 2019).

Tasarımcıların kullandığı genel metot, aşağıdan yukarıya olanıdır. Her tasarım, standart kapılar kullanılarak gerçekleştirilmektedir. Bu metodun kullanıldığı tasarımlarda, karmaşıklık arttıkça tasarım aynı oranda zorlaşmaktadır. Bu sebeple tasarımcılar, aşağıdan yukarıya metodu yerine yapısal hiyerarşik tasarım metodunu tercih ederler (Docplayer, 25 Nisan 2019).

Daha karmaşık tasarımlarda, tasarımcıların tercih ettiği metot yukarıdan aşağıya metodudur. Yukarıdan aşağıya tasarım metodu, test çabukluğu ve farklı teknolojilerde kolay tasarım değişikliği gibi avantajlar sağlamaktadır. Fakat yukarıdan aşağıya tasarımı takip etmek tasarımcılar için zor bir durumdur. Sonuç olarak tasarımcılar her iki metodu da tasarımlarında kullanarak çözüme ulaşmaktadırlar (Docplayer, 25 Nisan 2019).

## 2.4 VHDL Donanım Tanımlama Dili

VHDL, Verilog-HDL gibi bir donanım tanımlama dilidir. Bu dilin gelişimi 1970'lerde başlamıştır. 1987 yılında resmi olarak IEEE standardı olarak kabul edilmiştir. 1993 yılında son halini almıştır (Sarıtaş ve Karataş, 2015).

(sf. 1) VHDL'in özellikleri aşağıdaki gibidir;

- Tasarımlar hiyerarşili şekilde bileşenlerine ayrılabilir.
- Her tasarım elemanı iyi tanımlı bir arayüze ve hatasız davranışsal tanımlamaya sahip olmalıdır.
- Davranışsal tanımlama yapılırken algoritma veya gerçek donanım yapıları kullanılarak elemanların işlemi belirtilir.
- Uyumluluk, zamanlama ve darbe işareti denetimi modellenebilir. VHDL senkron ve asenkron ardışıl devre yapılarını gerçekleyebilir.
- Bir tasarımın lojik işlemleri ve zaman davranışının simülasyonunu yapılabilir (Iucoders, 25 Nisan 2019).

### 2.4.1 Donanım Tanımlama Biçimleri

Donanım tanımlama üç şekilde yapılmaktadır. Bunlar yapısal tanımlama, davranışsal tanımlama ve veri akışı yaklaşımıdır.

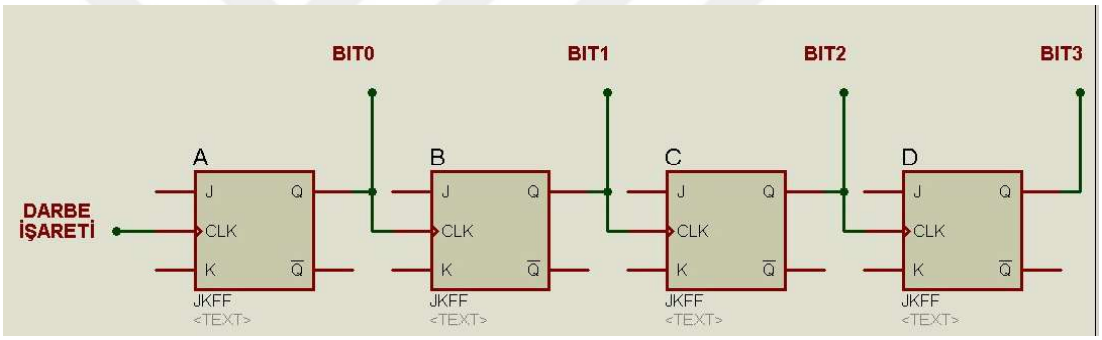
Sayısal bir modülün içyapısını oluşturan elemanlar lojik kapıların veya çeşitli lojik işlevi yerine getiren alt blokların birbirleri ile bağlantılarını dikkate alarak yapılan tanımlama yapısal tanımlamadır (Güneş ve Örs, 25 Nisan 2019).

Sistemden istenilen davranışı, lojik yapıdaki karşılığını düşünerek programlama dillerini andıran tarzda bir tanımlama çeşidine davranışsal tanımlama denmektedir (Güneş ve Örs, 25 Nisan 2019).

Veri akışı yaklaşımı; veri akışını tanımlamada temel blokların girişlerinin ve çıkışlarının devre içinde nasıl bağlanacağının tanımlamasıdır (Güneş ve Örs, 25 Nisan 2019).

# 3. SAYICI

Flip-Flop'ların, tetikleme işaretlerine göre belli bir durum dizisini tekrarlayacak şekilde bağlanmasıyla elde edilen devre bütünlerine sayıcı denilmektedir. Sayıcılar tetikleme işaretlerinin uygulanmasına ve sayma yönüne göre 2 grupta sınıflandırılmaktadırlar. Kullanıcılar sayıcıları istedikleri şekilde değerlendirebilirler. Örneğin bir kullanıcı sayıcı kullanarak sistemde ne kadar işlem yapıldığını görüp bu işlemin nasıl değerlendireceğine karar verebilir. Şekil 3'te, en değersiz bit bloğu A bloğu ve en değerli bit bloğu D bloğu olan4 bitlik örnek bir sayıcı blok diyagramı verilmiştir (Karaca, 19 Nisan 2019; T.C. MEB, 19 Nisan 2019).




**Şekil 3:**4 Bitlik Sayıcı Blok Diyagramı.

## 3.1 Tetikleme İşaretlerinin Uygulanmasına Göre

Tetikleme sinyallerinin flip-flop'lara uygulanış zamanına göre sayıcılar asenkron sayıcılar ve senkron sayıcılar olmak üzere ikiye ayrılmaktadırlar. Asenkron sayıcılarda sayma işlemi için kullanılan tetikleme sinyali ilk sıradaki flip-flop'a uygulanmaktadır. İlk sıradaki flip-flop'un çıkışlarından alınan sinyaller bir sonraki flip-flop'u tetiklemektedir. Bu şekilde her flip-flop'un çıkışından alınan sinyaller bir sonraki flip-flop'u tetikleyerek sayma işlemini gerçekleştirmektedir. Senkron sayıcılarda ise sayıcı sistemindeki her flip-flop aynı anda tek bir hat üzerinden tetiklenerek sayma işlemini gerçekleştirmektedir (Karaca, 19 Nisan 2019; T.C. MEB, 19 Nisan 2019).

### 3.1.1 Asenkron Sayıcı

Asenkronun kelime anlamı “eş zamanlı olmayan” demektir. Sayıcıyı oluşturan flip-flop’ların durum değiştirme zamanları birbirleri ile aynı değildir. Flip-flop’lar uygulanan her tetikleme sinyalinde durum değiştirmektedir (Karaca, 19 Nisan 2019; T.C. MEB, 19 Nisan 2019).

Bu sayıcılarda flip-flop’ların tetikleme uçlarına gelen darbe işaretleri aynı zamanda olmamaktadır. Flip-flop'ların tetikleme ucuna gelen her bilgide çıkış durumu değişmektedir. Örneğin darbe işareti ilk geldiği anda çıkış durumu lojik 1 ikinci geldiği anda durumu lojik 0 olmaktadır. Bu durumda flip-flop'un çıkışındaki işaret sistem darbe işaretinin iki katı periyotunda yeni bir darbe işareti oluşturmaktadır. Asenkron sayıcılardaki ilk flip-flop'un tetikleme ucuna sistemin mevcut darbe işareti uygulanmaktadır. Ardından gelen her flip-flop'un tetikleme ucu bir önceki flip-flop'un çıkışına bağlanmaktadır. Aslında bu durum asenkron sayıcıların bir dezavantajı olarak belirtilmektedir. Çünkü devrede kullanılan flip-flop’ların tetiklenmesinin bir önceki flip-flop’a bağlı olması sayıcının çalışma hızını etkilemektedir. Örneğin asenkron sayıcı devresinde beş adet flip-flop kullanılmış olsun; her bir flip-flop’un tetiklenip veriyi işleme süresi 20 ns (50MHz) olduğunu kabul edersek, son flip-flop’un tetiklenip veriyi işlemesi için geçen toplam süre 5x20=100 ns olacaktır.

Flip-flop'ların özelliklerine göre tetikleme uçları iki durumda çalışmaktadırlar. Darbe işaretinin yükselen kenarına ve lojik 1 olduğu duruma göre işlem yapan flip-floplar ve darbe işaretinin düşen kenar ve lojik 0 olduğu duruma göre işlem yapan flip-floplar (Karaca, 19 Nisan 2019; T.C. MEB, 19 Nisan 2019).

### 3.1.2 Senkron Sayıcı

Senkron’un kelime anlamı “eş zamanlı” demektir. Senkron sayıcılarda devrede bulunan bütün flip-flop’lar aynı anda tetiklenmektedir. Senkron sayıcıların çıkış durumları aynı anda değişmektedir (Karaca, 19 Nisan 2019; T.C. MEB, 19 Nisan 2019).

Senkron sayıcılar çalışma hızı açısından asenkron sayıcılara göre daha hızlı olmaktadırlar. Her durumda, kullanılan flip-flop'ların bir tanesinin gecikme süresi kadar gecikmesi olmaktadır. Senkron sayıcıların tasarımında, asenkron sayıcılara göre devre elemanları daha fazla kullanılmaktadır (Karaca, 19 Nisan 2019; T.C. MEB, 19 Nisan 2019).

## 3.2 Sayma Yönüne Göre

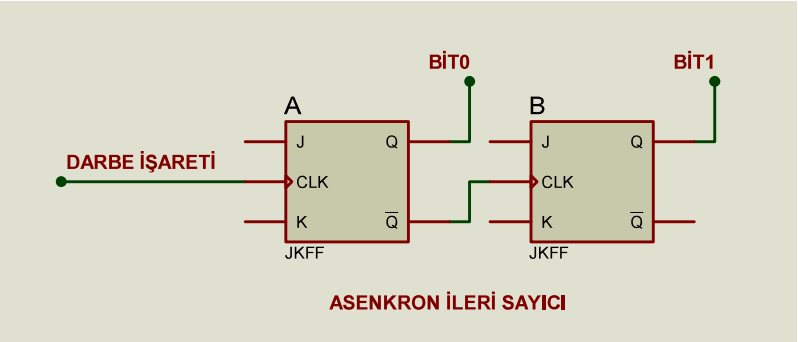
Sayıcılar, sayma yönüne göre üçe ayrılmaktadırlar. İleri sayıcılar, geri sayıcılar ve ileri/geri sayıcılar (Karaca, 19 Nisan 2019; T.C. MEB, 19 Nisan 2019).

### 3.2.1 İleri Yönlü Sayıcılar

İleri yönlü sayıcılar sayma işlemini arttırarak yapan sayıcılardır (Karaca, 19 Nisan 2019; T.C. MEB, 19 Nisan 2019).

#### 3.2.1.1 Asenkron İleri Sayıcı

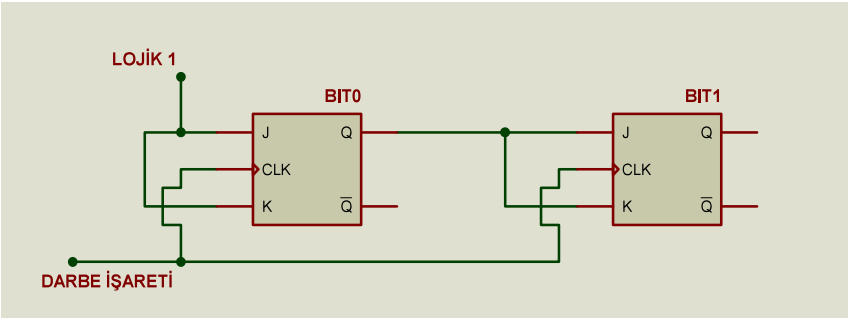
Şekil 4'de, iki bitlik asenkron ileri sayıcı devresi verilmiştir. A flip-flop'unun tetikleme ucuna sistem darbe işareti verilmiştir. A flip-flop'unun çıkışının değili (terslenmişi)(Q') B flip-flop'unun tetikleme ucuna bağlanmıştır. Flip-flop'ların Q çıkışları ise sırasıyla bit 0 ve bit 1 çıkışlarıdır (Karaca, 19 Nisan 2019; T.C. MEB, 19 Nisan 2019).




**Şekil 4:**Asenkron İleri Sayıcı.

### 3.2.1.2 Senkron İleri Sayıcı

Şekil 5'te, iki bitlik senkron ileri sayıcı devresi verilmiştir. Flip-flop'ların tetikleme uçlarına aynı anda sistem darbe işareti uygulanmaktadır. En değersiz bit'e ait olan flip-flop'un J ve K girişleri kısa devre yapılarak lojik 1 işareti sistem çalıştığı sürece verilmektedir. Ardından gelen flip-flop'ların J ve K girişleri yine kısa devre yapılarak, bir önceki flip-flop'un çıkışına bağlanmaktadır (Karaca, 19 Nisan 2019; T.C. MEB, 19 Nisan 2019).



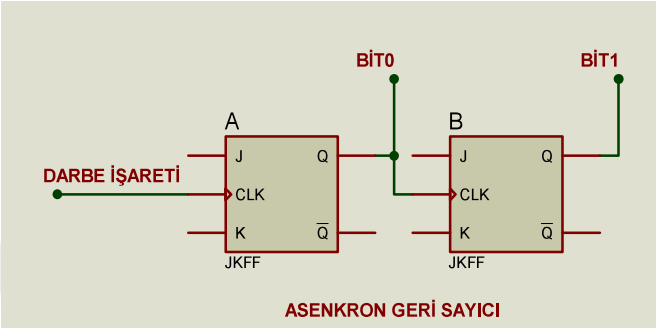

**Şekil 5:** Senkron İleri Sayıcı.

### 3.2.2 Geri Yönlü Sayıcılar

Geri yönlü sayıcılar sayma işlemini azaltarak yapan sayıcılardır (Karaca, 19 Nisan 2019; T.C. MEB, 19 Nisan 2019).

### 3.2.2.1 Asenkron Geri Sayıcılar

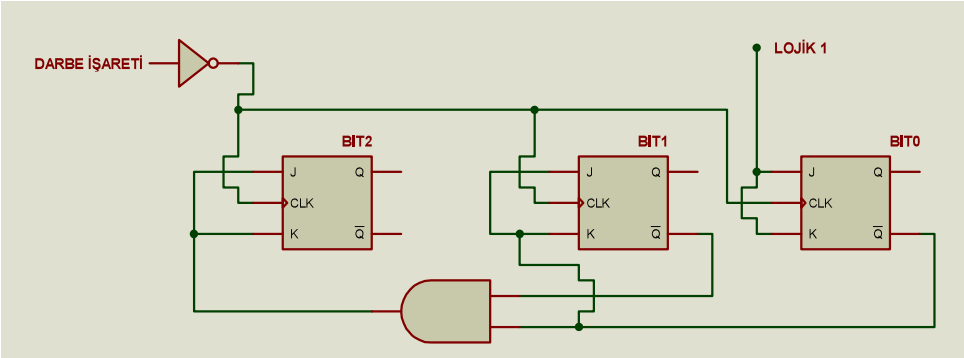
Şekil 6'da, iki bitlik asenkron geri sayıcı verilmektedir. A flip-flop'unun tetikleme ucuna sistem darbe işareti uygulanmaktadır. B flip-flop'unun tetikleme ucuna ise ileri sayıcıdan farklı olarak Q çıkışı bağlanmaktadır. Bit 0 ve bit 1 çıkışları flip-flop'ların Q çıkışlarından alınmaktadır (Karaca, 19 Nisan 2019; T.C. MEB, 19 Nisan 2019).




**Şekil 6:** Asenkron Geri Sayıcı.

### 3.2.2.2 Senkron Geri Sayıcılar

Şekil 7'de, üç bitlik senkron geri sayıcı verilmektedir. Flip-flop'ların tetikleme uçlarına aynı anda sistem darbe işaretinin tersi uygulanmaktadır. En değersiz bitin flip-flop çıkışının değili (Q') bir sonraki flip-flop'un J ve K girişlerine uygulanmaktadır. İkinci flip-flop'un çıkışının değili ise bir önceki flip-flop çıkışının değili ile lojik VE kapısı ile birleştirilip üçüncü flip-flop'un J ve K girişlerine bağlanmıştır. Her flip-flop bir bit değerini vermektedir. Eğer senkron geri sayıcı bit sayısı arttırılmak istenirse her flip-flop'un girişine kendinden önceki bütün flip-flop'ların çıkışlarının değilleri lojik VE kapısı ile birleştirerek, J ve K girişine bağlanmalıdır (Karaca, 19 Nisan 2019; T.C. MEB, 19 Nisan 2019).



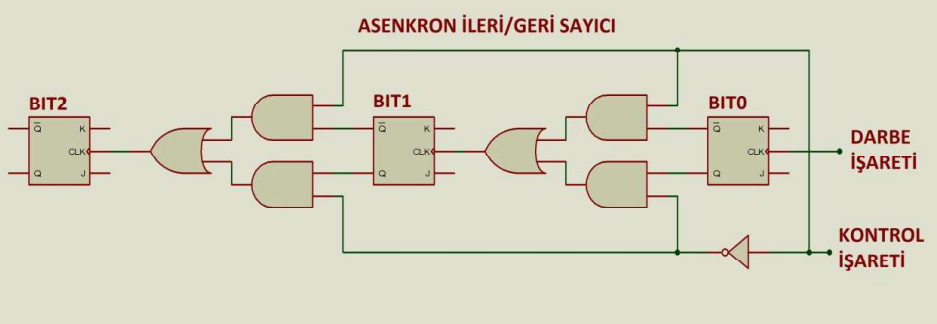

**Şekil 7:** Senkron Geri Sayıcı.

### 3.2.3 İleri/Geri Yönlü Sayıcılar

İleri/geri sayıcılar hem artarak hem azalarak sayma işlemi yapabilen sayıcılardır. İleri veya geri yönlü kullanılabilirler. Kullanıcı tasarladığı devrede ufak değişiklikler yaparak ileri veya geri veya hem ileri hem geri yönlü kullanabilir (Karaca, 19 Nisan 2019; T.C. MEB, 19 Nisan 2019).

#### 3.2.3.1 Asenkron İleri/Geri Sayıcılar

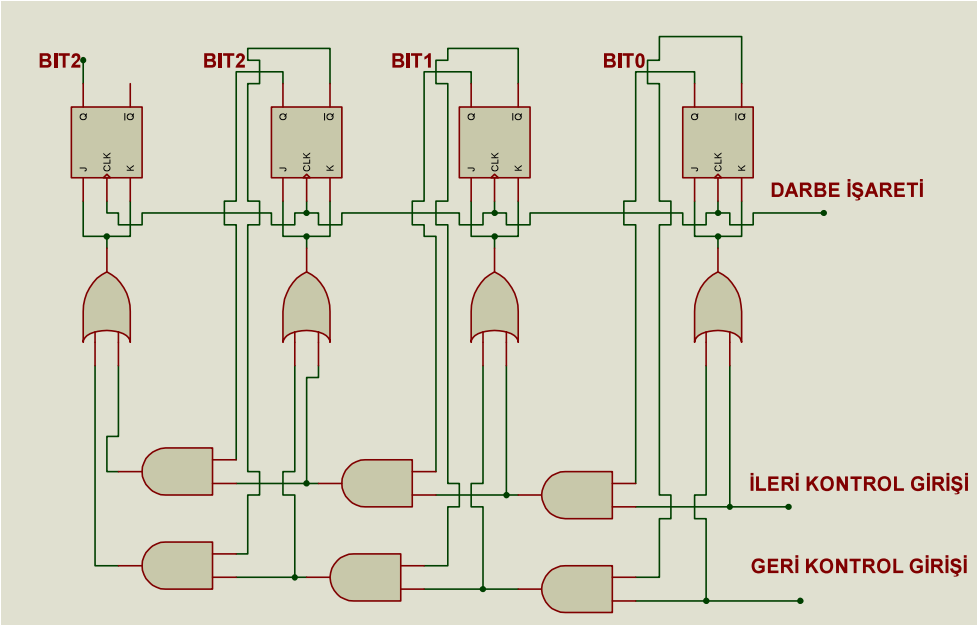
Şekil 8'de, üç bitlik bir asenkron ileri/geri sayıcı devresi verilmiştir Bu sayıcı devresinde bulunan kontrol işareti sayesinde kullanıcının isteğine göre ileri sayıcı, geri sayıcı veya hem ileri hem geri sayıcı olarak kullanılabilir. Bu sayıcılar tek taraflı sayma yapan asenkron sayıcılara göre daha fazla devre elemanından oluşmaktadır. Her flip-flop bloğunun arasına iki adet lojik VE kapısı ve bir adet lojik VEYA kapısı konulmaktadır. Kontrol işaretinin değili ile ilk flip-flop'un çıkışı birinci lojik VE kapısına girmektedir. Kontrol işareti ile ilk flip-flop'un çıkışının değili ikinci lojik VE kapısına girmektedir. Lojik VE kapılarının çıkışları ise lojik VEYA kapısına girmektedir. Lojik VEYA kapılarının çıkışı bir sonraki flip-flop'un tetikleme ucuna bağlanarak asenkron ileri/geri sayıcı elde edilmektedir (Karaca, 19 Nisan 2019; T.C. MEB, 19 Nisan 2019).




**Şekil 8:**Asenkron İleri/Geri Sayıcı.

### 3.2.3.2 Senkron İleri/Geri Sayıcılar

Şekil9'da, dört bitlik senkron ileri/geri sayıcı devresi verilmiştir. Sistem darbe işareti bütün flip-flop'ların tetikleme ucuna aynı anda verilmektedir. İleri veya geri saymasını sağlayacak iki adet kontrol girişi bulunmaktadır. İleri sayıcı için ileri kontrol girişi lojik 1 olmalıdır. Geri sayıcı için geri kontrol girişi lojik 1 olmalıdır. İki kontrol girişinin aynı anda lojik 1 veya lojik 0 olması durumunda sayma işlemi J-K flip-flop'ların özelliğinden dolayı gerçekleşememektedir (Karaca, 19 Nisan 2019; T.C. MEB, 19 Nisan 2019).



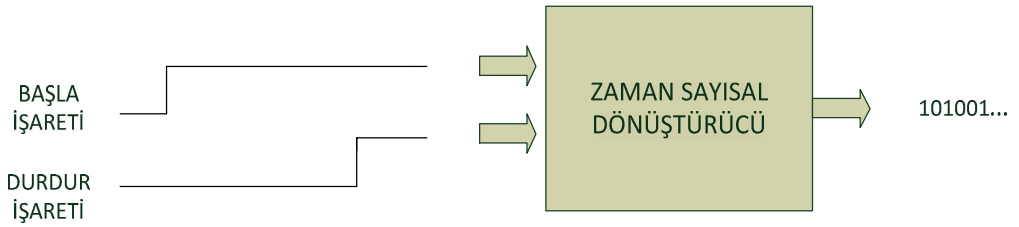

**Şekil 9:** Senkron İleri/Geri Sayıcı.

# 4. ZAMAN/SAYISAL DÖNÜŞTÜRÜCÜ (ZSD)

Zaman/sayısal dönüştürücü bir çeşit analog/sayısal dönüştürücü çeşididir. Analog bilgi devamlı veya kesintisiz bir bilgi demektir (Special minds, 4 Mayıs 2019). Örneğin literatürde insan gözünün saniyede 1000 kare fotoğraf algılama kapasitesinde olduğu görülmektedir. Çekilen videolarda ise genellikle saniyede 24 kare fotoğraf kullanılmaktadır. Bu şekilde bir saniyelik video 24 birimlik bir alan kaplarken eğer 1000 kare olsaydı 1000 birim alan kaplayacaktı. Yani analog bir bilgi 1000 birim iken bu bilgiyi sayısala dönüştürerek bilgi boyutu 24 birime inmiştir. Analog bilgiler sürekliliğini koruduğu sürece çok ciddi bir hafıza alanı ihtiyacı oluşturmaktadırlar. Günümüz teknolojisinde yüksek hafızalı ROM blokları olsa da bu bloklarda analog bilgi saklama sayısı yeterli gelmemektedir.

Bir analog bilgiyi saklayabilmek ve istenildiği gibi kullanabilmek için analog/sayısal dönüştürücüler kullanılmaktadırlar. Bu sayede analog bilginin sürekli olduğu durumlar değil ana hatlarının belli olduğu bilgiler kullanılarak hafıza yetersizlikleri için çözüme ulaşılır.

Bir başlama işareti ile bir durdurma işareti arasındaki zamanı sayısal bir işarete dönüştüren analog/sayısal dönüştürücüye zaman/sayısal dönüştürücü (ZSD) denmektedir. ZSD'ler elektronik ölçüm cihazlarında ve sinyal işlemede olayları tanımak ve gerçekleştikleri sürenin sayısal gösterimini sağlamak için kullanılmaktadırlar. ZSD 1 THz frekans ve daha fazlasında ölçüm yapmaya olanak sağlayabilir (Ahmed vd., 2016). Şekil 10'da, bir ZSD blok diyagramı verilmiştir.




**Şekil 10:** Zaman/Sayısal Dönüştürücü blok diyagramı.

ZSD'ler genellikle hassas ölçüm gerektiren yerlerde kullanılırlar. Kullanıldığı bazı yerler maddeler halinde yazılmıştır;

- Kapasitif sensör ölçüm devreleri (Xing vd., 2010),
- Nükleer tıp görüntüleme cihazları (Wang vd., 2017; Zhang vd., 2018),
- Parçacık fiziği (Wang vd., 2017; Balla vd., 2012; Zhang vd., 2018; Henzler, 2010),
- Yüksek enerji fiziği (Henzler, 2010),
- Biyo-kimyasal sensör kullanan ölçüm devreleri (Rezvanyvardom vd., 2014),
- Lazer ile mesafe ölçme (Van den Broek, 2012).

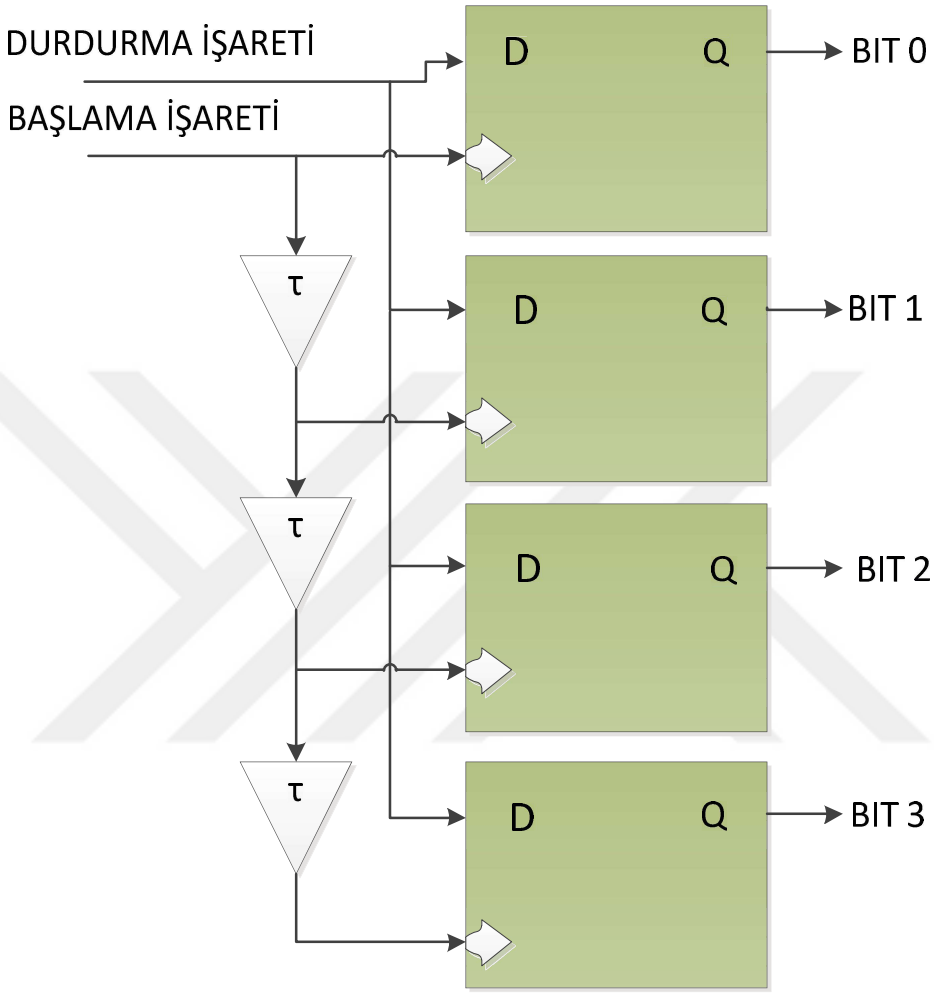
ZSD'leri oluşturabilmek için farklı metotlar mevcuttur. Aşağıda literatürde sık karşılaşılan metotlar maddeler halinde verilmiştir;

- Tümüyle paralel ZSD (Flash TDC),
- Vernier gecikme hatlı ZSD (Vernier delay line TDC),
- Vernier halka gecikmeli ZSD (Vernier ring TDC),
- Sayıcı Tabanlı ZSD (Counter Based TDC) (Van den Broek, 2012).

## 4.1 Tümüyle Paralel ZSD

Şekil 11'de, 4 bitlik bir tümüyle paralel ZSD yapısı verilmiştir. Sistemin başlama işareti, en değersiz bitin elde edildiği D tipi flip-flop'un tetikleme ucuna uygulanmıştır. Ardından her flip-flop'un tetikleme ucuna $\tau$ zamanlı gecikme blokları yerleştirilmiştir. Bu sayede her flip-flop'un tetikleme ucu bir önceki flip-flop'un tetikleme ucundan $\tau$ zamanı kadar gecikmiş olacaktır. Şekil 11'deki tümüyle paralel ZSD için başlama işareti "bit 0" için eş zamanlı çalışırken, "bit 1" için $\tau$ zamanı kadar gecikme, bit 2 için $2\tau$ zamanı kadar gecikme ve bit 3 için $3\tau$ zamanı kadar gecikme sağlanmıştır. Durdurma işareti $4\tau$ zamanından kısa olmalıdır. Durdurma işareti $4\tau$ zamanından daha geç sürede aktif olursa sistem baştan başlayacaktır. Bu durumda yanlış hesaplama yapılabileceğinden kullanıcı

seçim yaparken bunu göz önünde bulundurmak zorundadır (Van den Broek, 2012).

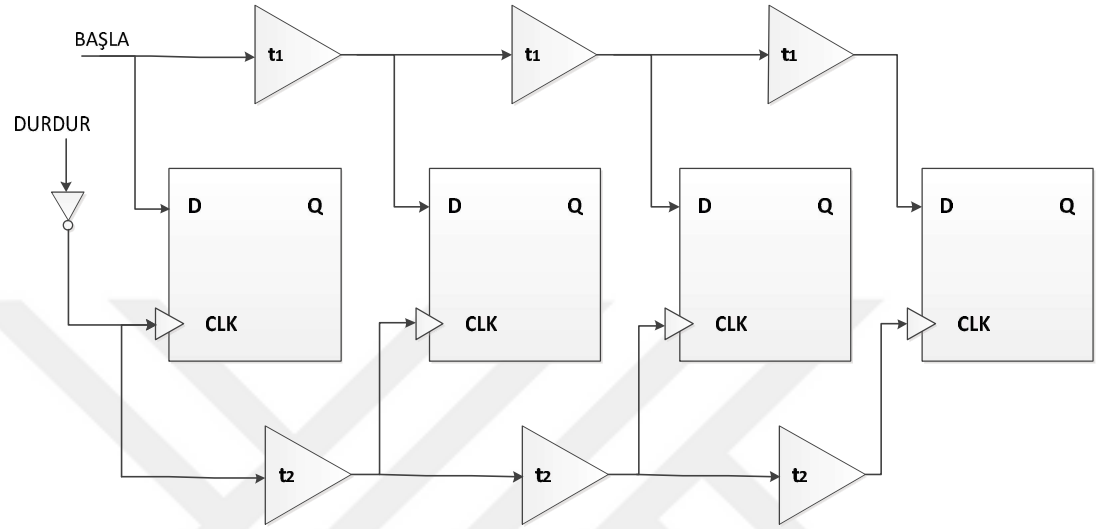



**Şekil 11:** 4 Bit Tümüyle paralel ZSD Bloğu.

### 4.2 Vernier'in Gecikme Hatlı ZSD'si

Şekil 12'te, 4 bitlik bir Vernier'in gecikme hatlı ZSD'si verilmiştir. Sistem bir başla işareti ile bir durdur işareti arasında geçen zamanı ölçmektedir. En değersiz bitin sahip olduğu D flip-flop'un ardından gelen her flip-flop'a işaretler gecikme blokları ile verilmiştir (Jovanovi´c ve Stoj^cev, 2009).

D Tipi flip-flop'unun enerji verildiği ilk anda "clk" yani tetikleme ucuna durdur işaretinin terslenmişi olan lojik 1 gelmektedir. Bu durumda Q çıkışı lojik 0 konumunda olmaktadır. D girişine başla işareti verildiğinde Q çıkışı lojik 1 olmaktadır. Durdur işareti sisteme verildiğinde Q çıkışı kendini lojik olarak tersleyerek lojik 0 konumuna geçmektedir.

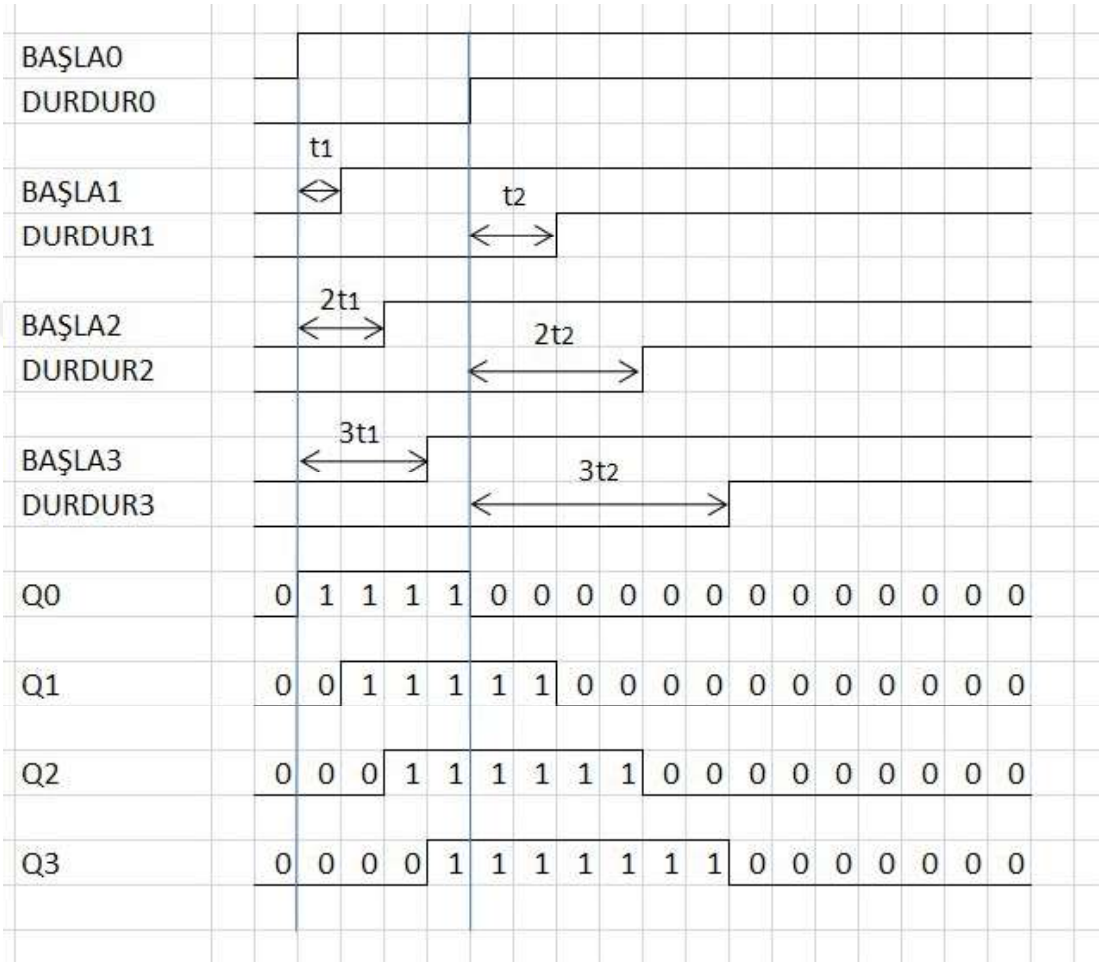



**Şekil 12:** Vernier'in gecikme hatlı ZSD'si.

Başla işaretinin bağlı olduğu hatta her gecikme bloğu $t_1$ kadardır. Durdurma işaretinin bağlı olduğu hatta ise her gecikme bloğu $t_2$ kadardır. Başlama işareti ile durdurma işaretinin arasında T zamanı kadar bir fark vardır. Bu fark ilk flip-flop'tan sonra her flip-flop'ta $\Delta t = t_1 - t_2$ kadar bir fark oluşturmaktadır. T zamanı boyunca ilk flip-flop hariç diğer flip-flop'lara enerji $\Delta t$'nin bağlı olduğu sıra sayısına (n) göre $n.\Delta t$ kadar gecikme ile gitmektedir (Jovanovi´c ve Stoj^cev, 2009).

Şekil 13'te,Vernier'in gecikme hatlı ZSD'sinin giriş çıkış işaretleri verilmiştir. Q0 ilk flip-flop çıkışı, Q1 ikinci flip-flop çıkışı, Q3 üçüncü flip-flop çıkışı ve Q4 dördüncü flip-flop çıkışıdır. BAŞLA0 ile DURDUR0 ilk flip-flop'un giriş işaretleridir. Ardından sırasıyla BAŞLA1-DURDUR1, BAŞLA2-DURDUR2 ve BAŞLA2-DURDUR2 işaretleri flip-flop'lara girmektedir. Her ikili işaret arasında "$T-n.\Delta t$" zamanı kadar fark oluşmaktadır.

Vernier'in gecikme hatlı ZSD'sinin avantajı hızlı olmasıdır. Δt her ne kadar küçük olsa da bu durum ZSD'nin çıkışında hata payları oluşturmaktadır. Dolayısıyla Vernier'in gecikme hatlı ZSD'sinin bit sayısı arttıkça hata oranı artmaktadır. Bu durum kullanıcıların hassas ölçüm yapması gerektiği alanlarda istenmeyen bir durumdur.

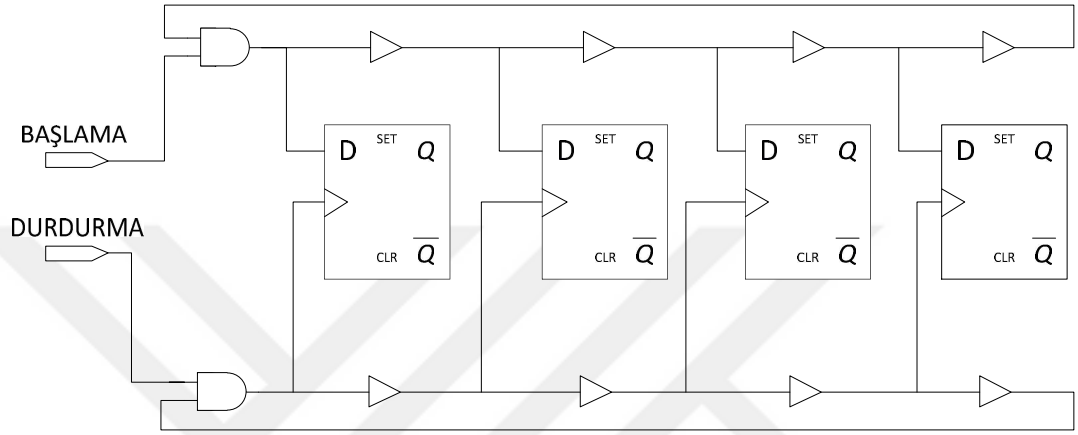



**Şekil 13:** Vernier'in gecikme hatlı ZSD'sinin çıkış işaretleri.

### 4.3 Vernier'in Halka Gecikmeli ZSD'si

Vernier'in halka gecikmeli ZSD'si, Vernier'in gecikme hatlı ZSD'sinden faklı olarak başlama ve durdurma işaretlerinin son gecikme bloklarının çıkışlarının olduğu hattın başına lojik VE kapısı ile birleştirilmesiyle elde edilmiştir. Şekil 14'de, 4 bitlik Vernier'in halka gecikmeli ZSD'si verilmiştir.

Flip-flop'lar, tetikleme uçlarına durdurma sinyali, giriş kısımlarına ise başlama işareti verilerek çalıştırılırlar. Vernier'in halka gecikmeli ZSD'si hem iyi bir zaman çözünürlüğü performansı hem de geniş bir faz algılama aralığı olması için geliştirilmiştir. Son gecikme bloğunun tekrar ilk bloğa bağlanması gecikme aşamasını büyük ölçüde azaltmakta olup, kullanılan alanı ve güç maliyetini düşürmektedir (Yu vd., 2010).

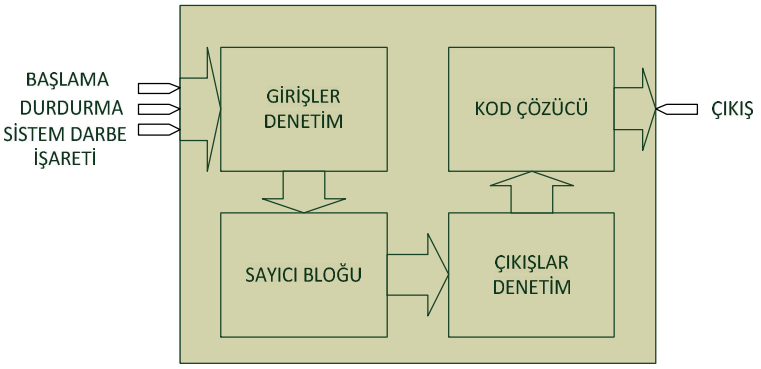



**Şekil 14:** 4 Bitlik Vernier'in halka gecikmeli ZSD'si.

Vernier'in halka gecikmeli ZSD'si frekansı düşük olan işlemlerde doğru çalışmamaktadır. ZSD'ler çok hassas zaman aralığını ölçmektedirler. Frekansı düşük işlemler için kullanılamazlar. Bunun nedeni gecikme bloklarının işlem hızlarının çok düşük sürelerde olmasından kaynaklanmasıdır (Yu vd., 2010).

## 4.4 Sayıcı Tabanlı ZSD

Bir başlama işareti ile durdurma işareti arasında geçen zamanı ölçmeye yarayan dönüştürücü takımına ZSD denmektedir. ZSD ile çok hassas zaman aralıkları ölçülebilmektedir. ZSD oluşturmak için birden fazla metot bulunmaktadır. Bu metotlar tasarımcının amacına göre değişmektedir. Sayıcı tabanlı bir ZSD'nin blok diyagramı şekil 15'te verilmiştir.




**Şekil 15:** Sayıcı tabanlı ZSD blok diyagramı.

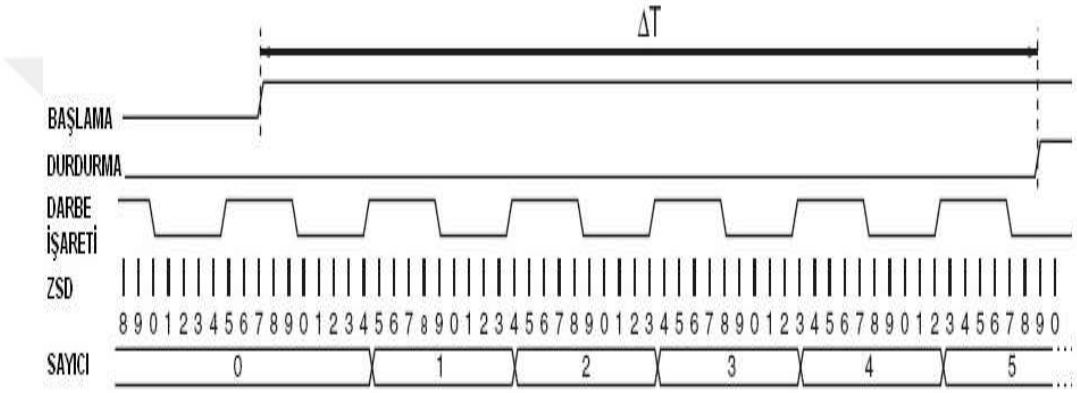
Sayıcı tabanlı ZSD dört kısımdan oluşmaktadır. İlk kısım giriş denetimlerinin sağlandığı alandır. Burada hangi girişin nereye, nasıl bağlanacağına karar verilmektedir. Bu karar mekanizmasından sonra girişler ikinci kısımda olan sayıcı bloğuna aktarılmaktadır. Giriş işaretlerinin durumuna göre sayıcı bloğu işlem yapmaktadır. İşlem sonucu üçüncü kısım olan çıkış denetim bloğuna giriş yapmaktadır. Burada sistem darbe işareti olduğu sürece sayıcı çıkışındaki bilgi kod çözücü bloğuna aktarılmaktadır. Kod çözücü bloğu sayıcı bloğundan gelen bilgileri kullanıcı tarafından istenilen bilgi işaretine/işaretlerine dönüştürerek çıkış işareti sağlamaktadır. Kod çözücü bloğu tasarımcının hangi sayma tabanına ihtiyacı olduğu belirlenerek seçilmektedir.

ZSD'ler çok hassas zaman aralıklarını ölçmek için kullanılmaktadırlar. Dolayısıyla büyük aralıkları ölçmek için kullanıldıklarında hatalı çalışmaktadırlar. Sayıcı tabanlı ZSD sistemin darbe işaretinin frekansına ve sayıcı bloğunun bit sayısına göre çalışmaktadır (Dadouche vd., 2015). Bu durum bize istediğimiz aralıktaki zamanı ölçebilmemize olanak sağlamaktadır. Sistem darbe işaretinin periyodu arttırılırsa, arttırılan oranda ZSD'nin hassasiyeti azalmaktadır. Sistem darbe işaretinin periyodu azalttırılırsa, azalttırılan oranda ZSD'nin hassasiyeti artmaktadır. Kullanıcı sadece sistem darbe işaretinin frekansını ve sayıcı bloğunun bit sayısını değiştirerek sayıcı tabanlı ZSD'yi istediği her alanda kullanabilmektedir (Keränen, 2016).

Bir başlama işareti sisteme verildiğinde, sayıcı bloğu birer birer artarak saymaya başlamaktadır. Bu sayma işlemi bir durdurma işareti gelene kadar devam etmektedir. Bu sayıcı bloğu sistem darbe işaretinin sahip olduğu periyot ile paralel

olarak çalışmaktadır. Durdurma sinyali geldiğinde periyot ile sayıcının son saydığı rakam çarpılarak sinyalin ne kadar süre geldiği hesaplanmaktadır.

Şekil 16'da, sayıcı tabanlı ZSD için örnek sinyaller verilmiştir. Bu örneğe göre hesaplama yapılacak olunursa, sayıcı durdurma işareti geldiğinde en son 5 saymıştır. Sistem darbe işaretinin frekansı 50 MHz olarak kabul edilirse, darbe işaretinin periyodu 20 ns olmaktadır. Şekil 17'de görülen "ΔT", başlama işareti ile durdurma işareti ile arasındaki zaman farkıdır. Bu zaman farkı ΔT=5*20 ns = 100ns şeklinde hesaplanmaktadır.




**Şekil 16:** Sayıcı tabanlı ZSD giriş – çıkış işaretleri.

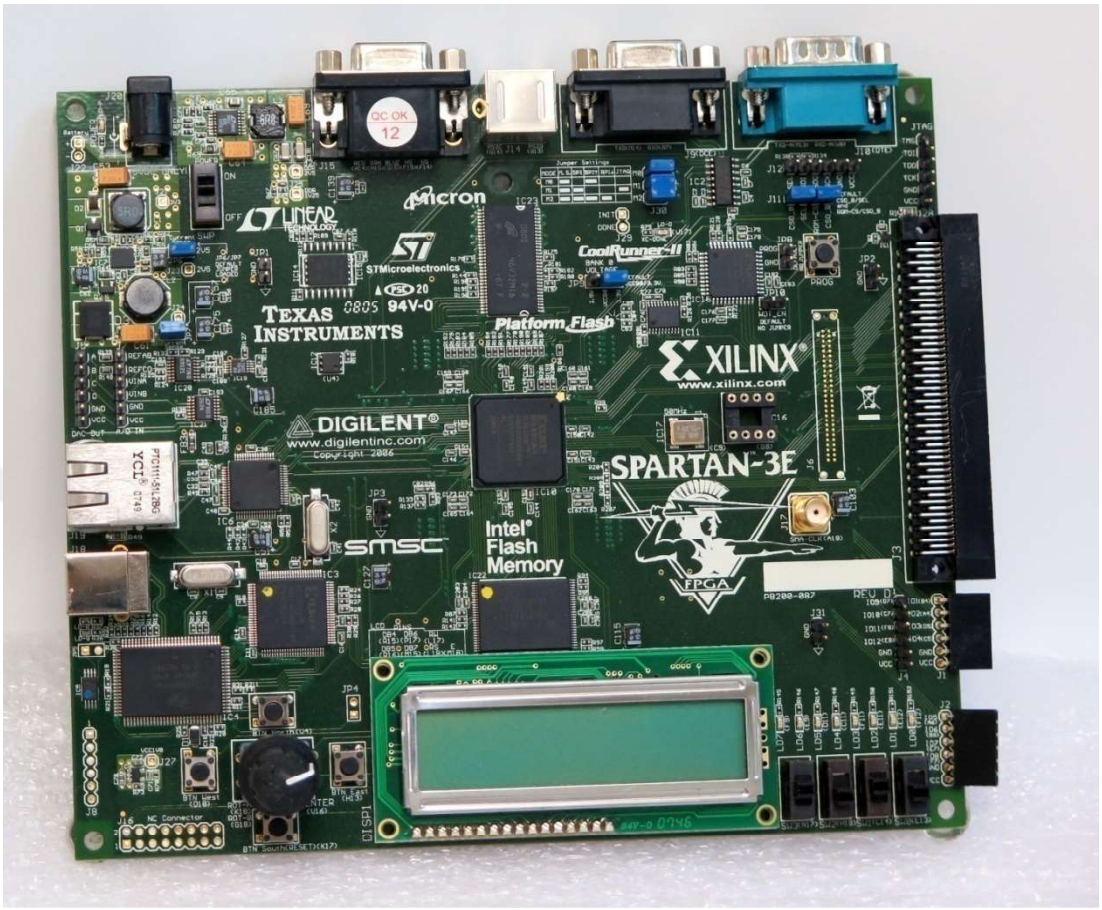
# 5. SPARTAN 3E BAŞLANGIÇ KARTI

Spartan-3E FPGA ailesi, elektronik uygulamalarıyla uğraşan kullanıcıların geniş kapsam ve ucuz maliyet gibi ihtiyaçlarını karşılamak için özel olarak tasarlanmıştır. Beş üyeli aileden oluşan Spartan-3E ailesi 100,000 ile 1,6 milyon sistem kapısı arasında değişen yoğunluklar sunmaktadır. Spartan-3E ailesi, daha önceki Spartan-3 ailesinin başarısını, lojik giriş / çıkış miktarını artırarak, mantık hücre başına maliyetini önemli ölçüde azaltarak geliştirmiştir. Bu yeni özellikler sistem performansını iyileştirmiş ve yapılandırma maliyetini düşürmüştür. Spartan-3E FPGA ailesinin donanımları, gelişmiş 90 nm işlem teknolojisi ile birlikte, programlanabilir mantık endüstrisinde yeni standartlar belirleyerek, daha önce mümkün olandan daha fazla işlevsellik ve bant genişliği sunmaktadır. Spartan-3E FPGA ailesi son derece düşük maliyetli olmalarından dolayı, geniş bantlı erişim, ev ağı, ekran / projeksiyon ve dijital televizyon ekipmanı gibi çok çeşitli elektronik uygulamalarıyla uğraşan kullanıcılar için idealdir (Xilinx, 2007, 25 Nisan 2019).

FPGA teknolojisini anlamak için öncelikle ASIC teknolojisini anlamak gerekmektedir. ASIC, tasarıma özel olarak üretilmiş tümleşik devre yongalarıdır. Programlanabilir mantık cihazları veya standart mantık devrelerine göre üstün performans sergilerler. Bunun sebebi tek bir amaç için üretilmiş olmasıdır. Çok küçük bir alanda ve çok az enerji tüketecek şekilde üretilebilir (Kara, 7 Mayıs 2019). ASIC teknolojisinin en büyük dezavantajı, birden fazla yongaya ihtiyaç duyulduğunda tasarım ve üretim için oldukça maliyetli olmasıdır.

Spartan-3E FPGA ailesi, programlanmış ASIC maskelerine göre üstün bir alternatiftir (Xilinx, 2018, 7 Mayıs 2019). FPGA'lar yüksek başlangıç maliyetinden, uzun gelişim döngülerinden ve geleneksel ASIC'lerin içsel esnekliğinden kaçınırlar. Ayrıca FPGA’lar, ASIC'ler gibi donanım değişikliği imkânsızlığını gerektirmediğinden alanda daha iyi tasarım yapılmasını mümkün kılmaktadırlar.

Şekil 17'de, Spartan 3E başlangıç kartı gösterilmiştir. Bu tez çalışmasında Spartan 3E kartının XC3S1600E serisi kullanılmıştır.




**Şekil 17:** Spartan 3E başlangıç Kartı.

## 5.1 SPARTAN 3E Başlangıç Kartı Özellikleri

Kullanıcılar tasarım yapacağı kartı seçerken kartın veri dokümanına göre karar vermektedirler. Kullanılan Spartan 3E başlangıç kartının başlıca özellikleri dört başlıkta verilmiştir (Xilinx, 2007, 25 Nisan 2019).

### 5.1.1 SPARTAN 3E FPGA Gömülü İşleme İşlevleri

MicroBlaze sanal bir işlemcidir. Çekirdek (core) adı verilen blokların FPGA içerisinde birleştirilmesi ile oluşturulmaktadır. MicroBlaze sanal bir

işlemci olduğu için FPGA içerisinde birden fazla oluşturulabilir ve her işlemciye istenildiği kadar çekirdek (Uart, ethernet, pci vb) eklenebilir (FPGA, 14 Nisan 2019).

PicoBlaze, Xilinx'in kendi FPGA'leri için sunduğu düşük performans / düşük maliyet sınıfı 8 bit'lik soft (FPGA'ye gömülebilen) işlemcisidir (FPGA, 14 Nisan 2019).

ChipScope, tasarımımızı FPGA yongasına yükledikten sonra, ürün çalışırken tasarımda kullanılan tüm veri yolları (bus) ve sinyalleri izlememize olanak sağlayan bir programdır (FPGA, 14 Nisan 2019).

MicroBlaze geliştirme kiti kartı, Spartan-3E FPGA ailesinin benzersiz özelliklerini vurgular ve gömülü işleme uygulamaları için uygun bir geliştirme kartı sunar. Spartan-3E kartı aşağıdaki özellikleri vurgulamaktadır (Xilinx, 2007, 25 Nisan 2019);

- Paralel NOR Tümüyle paralel konfigürasyonu,
- Paralel NOR Flash PROM'dan MultiBoot FPGA konfigürasyonu,
- SPI seri Flash yapılandırması,
- Gömülü gelişim,
- MicroBlaze 32-bit gömülü RISC işlemci,
- PicoBlaze ™ 8-bit yerleşik denetleyici,
- DDR bellek arayüzleri,
- 10-100 Ethernet,
- UART (Xilinx, 2018, 7 Mayıs 2019).

MicroBlaze geliştirme kiti kartı diğer Spartan geliştirme kartlarına göre daha gelişmiş ve karmaşıktır. MicroBlaze geliştirme kiti kartı, MicroBlaze yerleşik işlemcisinin ve Xilinx Dahili Geliştirme Kitinin (EDK) temel özelliklerini gösterir (Xilinx, 2007, 25 Nisan 2019).

### 5.1.2 Pimler Ve Özellikleri

Xilinx XC3S1600E Spartan-3E FPGA kartı aşağıdaki özellikleri içermektedir;

- 250'ye kadar kullanıcı tanımlı giriş/çıkış pimleri,
- FPGA için toplam 320 pim,
- 33.000'den fazla mantık hücresi,
- İki adet 4 Mbit tümüyle paralel yapılandırmalı PROM,
- 64 - makro hücre XC2C64A harika kanallı CPLD,
- 64 MByte (512 Mbit) DDR SDRAM, 16x veri arabirimi, +100 MHz,
- 16 MByte (128 Mbit) paralel NOR Tümüyle paralel (Intel StrataFlash),
- FPGA yapılandırma depolaması,
- MicroBlaze kod depolama / tekrar kullanma,
- 16 Mbits SPI seri Tümüyle paralel (STMicro),
- 2 x 16 LCD ekran,
- PS / 2 fare veya klavye bağlantı noktası,
- VGA ekran bağlantı noktası,
- 10/100 Ethernet fiziksel arayüzü,
- İki adet 9 pimli RS-232 portu (DTE ve DCE tarzı),
- Dâhili USB tabanlı FPGA / CPLD indirme / hata ayıklama arayüzü,
- 50 MHz (tümleşik) ve 66 MHz (değiştirilebilir 8 pimli DIP soketi) darbe işareti üreteci,
- Bit akımı kopya koruması için SHA-1 1-telli seri EEPROM,
- Kullanıcı tanımlı Giriş / Çıkış özellikli 40 pimli FX2 geniş konektör,
- Üç adet 6-pimli konektör,
- Dört çıkışlı, SPI tabanlı Sayısal Analog Dönüştürücü (DAC),
- Programlanabilir kazançlı ön amplifikatöre sahip iki girişli, SPI tabanlı Analog-Sayısal Dönüştürücü (ADC),
- ChipScope hata ayıklama portu,

- Düğme şaftlı döner kodlayıcı,
- Sekiz adet led,
- Dört adet sürgülü anahtar,
- Dört adet düğmeli anahtar,
- SMA darbe işareti girişi (Xilinx, 2007, 25 Nisan 2019).

### 5.1.3 Yapılandırma Yöntemleri

Tipik bir FPGA uygulaması, yapılandırma görüntülerini saklamak için geçici olmayan bir bellek kullanmaktadır. Yeni Spartan-3E yeteneklerini göstermek için, MicroBlaze geliştirme kiti kartı ile birlikte iyi çalışması gereken üç farklı yapılandırma belleği kaynağına sahiptir. Ekstra yapılandırma fonksiyonları, başlangıç kiti kartını tipik Spartan-3E uygulamalarında daha karmaşık hale getirmektedir. Başlangıç kiti kartı ayrıca bir adet USB tabanlı JTAG programlama ara yüzü içermektedir. Çip üzerindeki devre, cihaz programlama deneyimini kolaylaştırmaktadır. Tipik uygulamalarda, JTAG programlama donanımı, yerleşik veya Xilinx Platform USB kablosu gibi ayrı bir programlama modülünde bulunmaktadır. Bu USB portu sadece programlama içindir ve bağımsız bir USB ara yüzü olarak kullanılamamaktadır (Xilinx, 2007, 25 Nisan 2019).

### 5.1.4 Spartan 3E Uygulamaları İçin Besleme Gerilimi

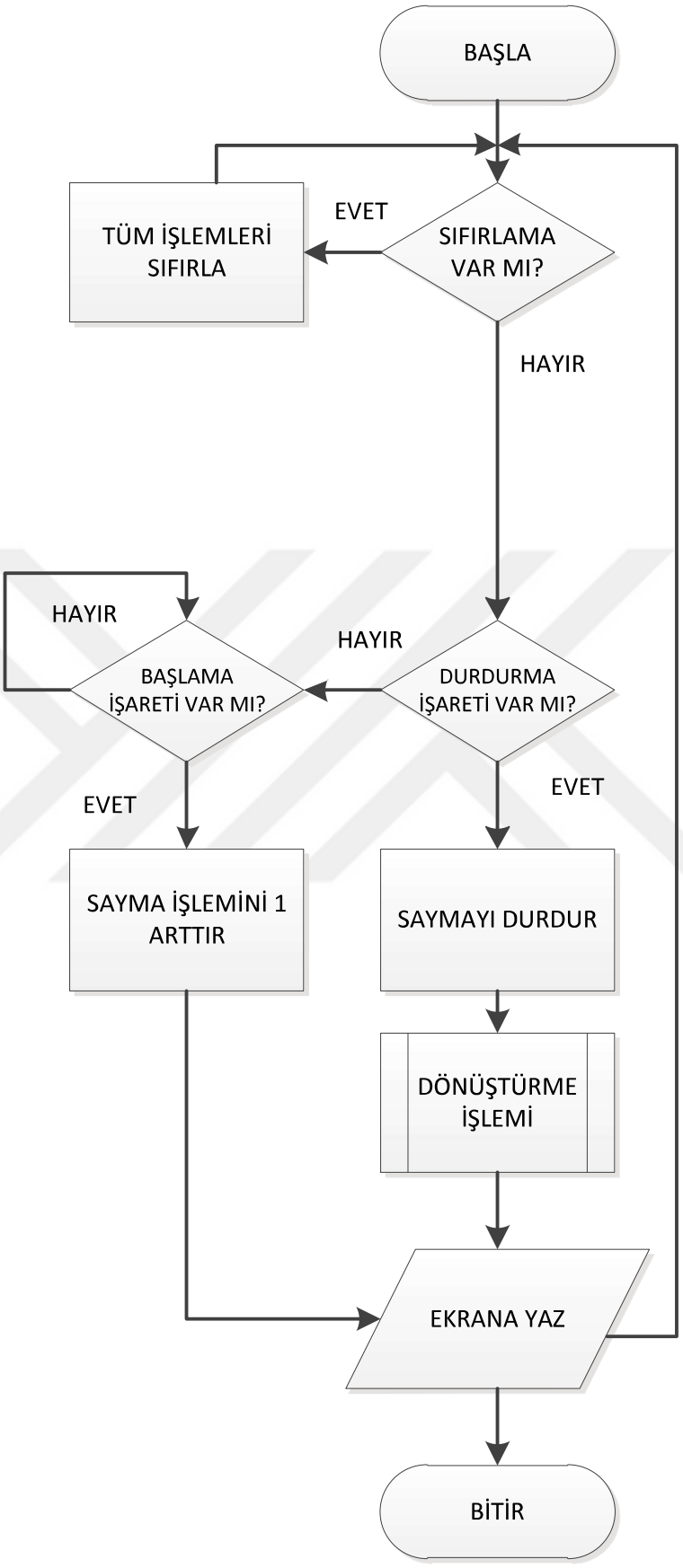
MicroBlaze geliştirme kiti kartı, özellikle Spartan-3 ve Spartan-3E FPGA'ları güçlendirmek için Texas Instruments tarafından geliştirilen TPS75003 kodlu üç çıkışlı bir voltaj regülatörü kullanmıştır. Bu regülatör çoğu bağımsız FPGA uygulaması için yeterlidir. Bununla birlikte, başlangıç kiti kartı kendi yüksek akım kaynağını oluşturabilen DDR SDRAM'ı içermektedir. Benzer şekilde, USB tabanlı JTAG yüklemesi için bir adet 1.8V besleme bulunmaktadır (Xilinx, 2007, 25 Nisan 2019).

# 6. FPGA'DA SAYICI TABANLI ZSD TASARIMI

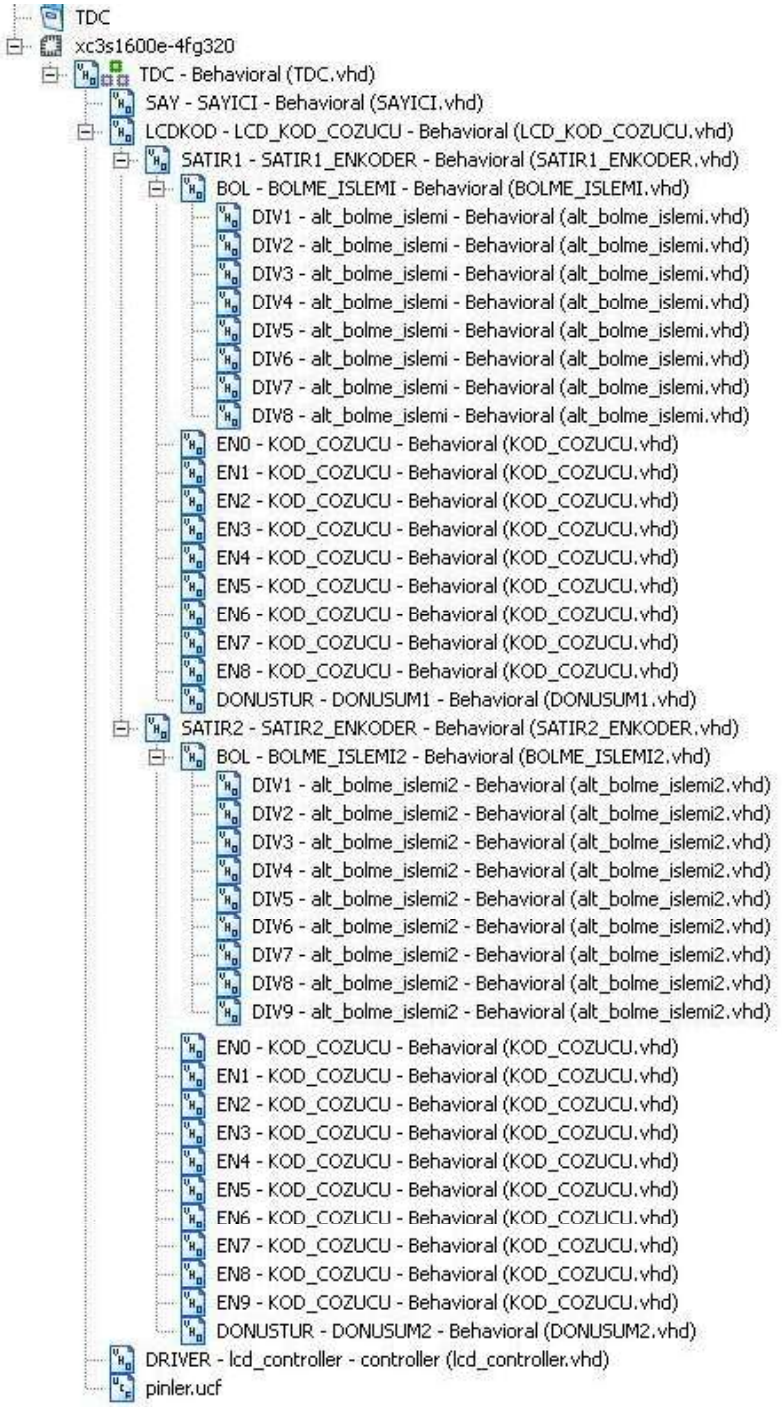
## 6.1 Tasarım

Bu tez çalışmasında Spartan 3E başlangıç kartında bir başlama ve bir durdurma işaretinin arasında geçen zamanın sayısal olarak çıkış vermesini sağlayan ZSD tasarlanmıştır. Bu çıkış işareti Spartan 3E başlangıç kartındaki LCD ekrana verilmiştir. Başlama ve durdurma işaretleri Spartan 3E başlangıç kartındaki giriş/çıkış pimleri vasıtasıyla harici kaynaktan alınmıştır.Başlama ve durdurma işaretleri 4 farklı sistem frekansında ve üç farklı doluluk-boşluk oranında denenmiştir. Seçilen sistem frekansları sırasıyla 20MHz (50 ns), 40MHz (25 ns), 50MHz (20 ns) ve 66MHz (15ns)'dir. Doluluk-boşluk oranları sırasıyla %25-%75, %50-%50 ve %75-%25'dir.

Darbe üreteçleri Spartan 3E başlangıç kartı sisteminde darbe işareti oluşturmaktadır. Bu darbe işareti devamlı olarak üretilmektedir. Başlama işareti geldiğinde Spartan 3E başlangıç kartında tasarlanan ZSD'nin içinde gömülü olan sayıcı bloğu, üretilen darbe işaretine bağlı olarak saymaya başlayacaktır. Sayıcı bloğu, sistem frekansına bağlı olarak saniyede 20 milyon ile 66 milyon arasında bir işaret üretmektedir. Bu esnada Spartan 3E başlangıç kartının LCD ekranında, üretilen işaret gözlemlenmektedir. Ancak işaret çok hızlı olduğu için LCD ekran üzerindeki çıkış işareti gözle algılanamamaktadır. Durdurma işareti geldiğinde ZSD'nin içinde gömülü olan sayıcı bloğu saymayı durduracaktır. Durdur işareti ile birlikte çıkış işareti artık gözle algılanabilir olacaktır. Şekil 18'de tasarlanan çalışmanın algoritması, şekil 19'da VHDL dilinde yazılan kodların hiyerarşisi gösterilmektedir.




**Şekil 18:** Yapılan çalışmanın algoritması.




**Şekil 19:**VHDL Kod Hiyerarşisi.

VHDL dilinde oluşturulan kodlar ekler kısmının EK A bölümünde verilmiştir. Ek A.1'de oluşturulan ZSD'nin ana kodu verilmiştir. Çift çizgi ile başlayan cümleler kullanıcının kodlara tekrar baktığında rahat bir şekilde hatırlaması için notlardır. Bu notlar olmasa da program çalışmaktadır.

Ek A.2'de 32 bit sayıcı tasarlanmıştır. Her sistem darbe işaretinin yükselen kenarında ve lojik 1 olduğu durumda sayıcı çıkışı 1 artacaktır. Sistemde darbe işareti oluşturabilmek için Spartan 3E başlangıç kartında üç faklı seçenek bulunmaktadır. Bunlar; kart üzerinde tümleşik olarak bulunan 50 MHz'lik sabit darbe üreteci, kullanıcının farklı frekanslarda çalışmasına imkân sağlayan 4-8 bacaklı darbe üreteci yuvası ve sabit darbe üreteçlerin haricinde kullanıcının farklı sistemlerden darbe işareti verebileceği darbe üreteci pimi mevcuttur.

Ek A.3'deki kod takımında LCD ekranın üst ve alt satırları için enkoder tasarlanmıştır. Bu enkoder, sayıcı bloğundan çıkan ikili sayı sistemini LCD'de gözlemlemek için karakter verilerine dönüştürmektedir.

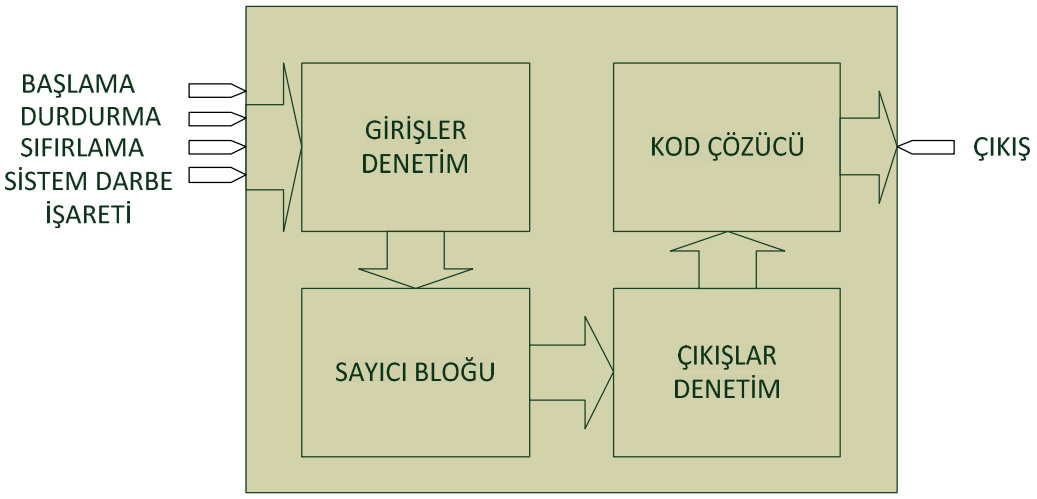
Ek A.4'de LCD'nin üst satırı için yardımcı enkoder kodu verilmiştir. Ek A.5'de LCD'nin alt satırı için yardımcı enkoder kodu verilmiştir. Ek A.6'da satırların enkoderi için bölme işlemini yapacak program bulunmaktadır. Ek A.7'de bölme işlemini yapacak programın yardımcı alt kodu bulunmaktadır.

Ek A.8'de LCD ekranın veri dokümanına göre sağlıklı bir şeklide çalışması sağlayan kod yazılmıştır. Bu kodun sağlıklı çalışması için sistemde darbe üreteci tarafından üretilen darbe işaretinin değerinin *"**CONSTANT freq: INTEGER :=**"* kod parçasının yanına MHz cinsinden yazılması gerekmektedir.

Oluşturulan kodu karta aktarmadan önce pim tanımlamasının kesinlikle yapılması gerekmektedir. Pim tanımlamaları her kartın kendine özel olan veri dokümanına göre hazırlanmaktadır. Ek A.9'da Spartan 3E başlangıç kartı üzerindeki pimlerin tanımlamaları yazılmıştır.

## 6.2 Simülasyon Sonuçları

Tasarlanan ZSD şekil 20'de verilmiştir. Bu tasarım 4 farklı sistem frekansı ile denenmiştir. Farklı frekanslarda test yapılabilmesi için yazılan kodlarda bazı değişiklikler yapılması gerekmektedir. Bu değişikliklerin yapılması gereken kısımlar LCD kod takımı ve pim tanımlamaları kodlarındadır. Her bir kod takımı doğru şekilde yazıldıktan sonra simülasyon testleri gerçekleştirilmiştir.



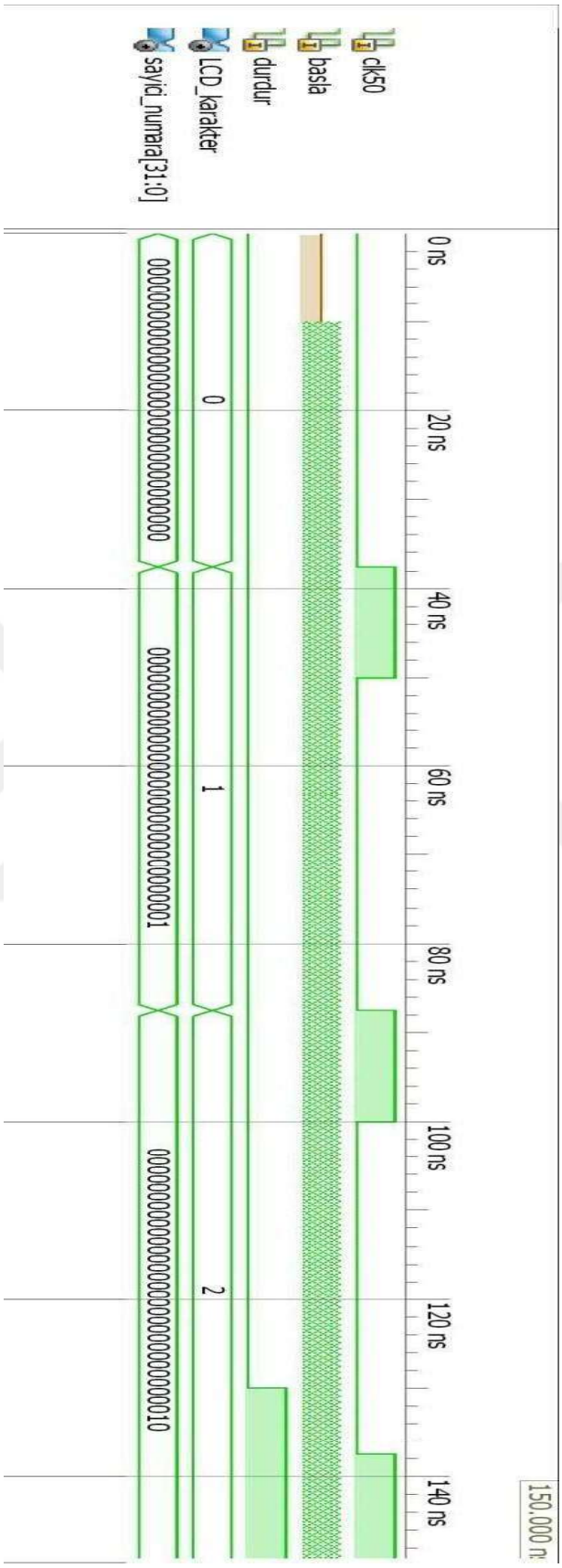

**Şekil 20:** Tasarlanan ZSD.

Spartan 3E başlangıç kartı için 4 farklı darbe üretecinin her birinin ürettiği darbe işareti için 3 farklı doluluk - boşluk oranları seçilip test edilmiştir. Bu doluluk - boşluk oranları %25-%75, %50-%50, %75-%25 olarak seçilmiştir. Toplamda 12 farklı simülasyon sonucu elde edilmiştir.

"clk50" isimli sinyal darbe üretecinin sistemde ürettiği darbe işaretidir, "basla" isimli giriş sinyali kullanıcının tanımladığı başlama işaretidir. "durdur" isimli giriş sinyali kullanıcının tanımladığı durdurma işaretidir. "LCD_karakter" isimli çıkış sinyali LCD ekranda görülen karakter işaretidir, "sayici_numara[31:0]" isimli sinyal sayıcı bloğunun ikili sayma sistemindeki çıkış durumudur.

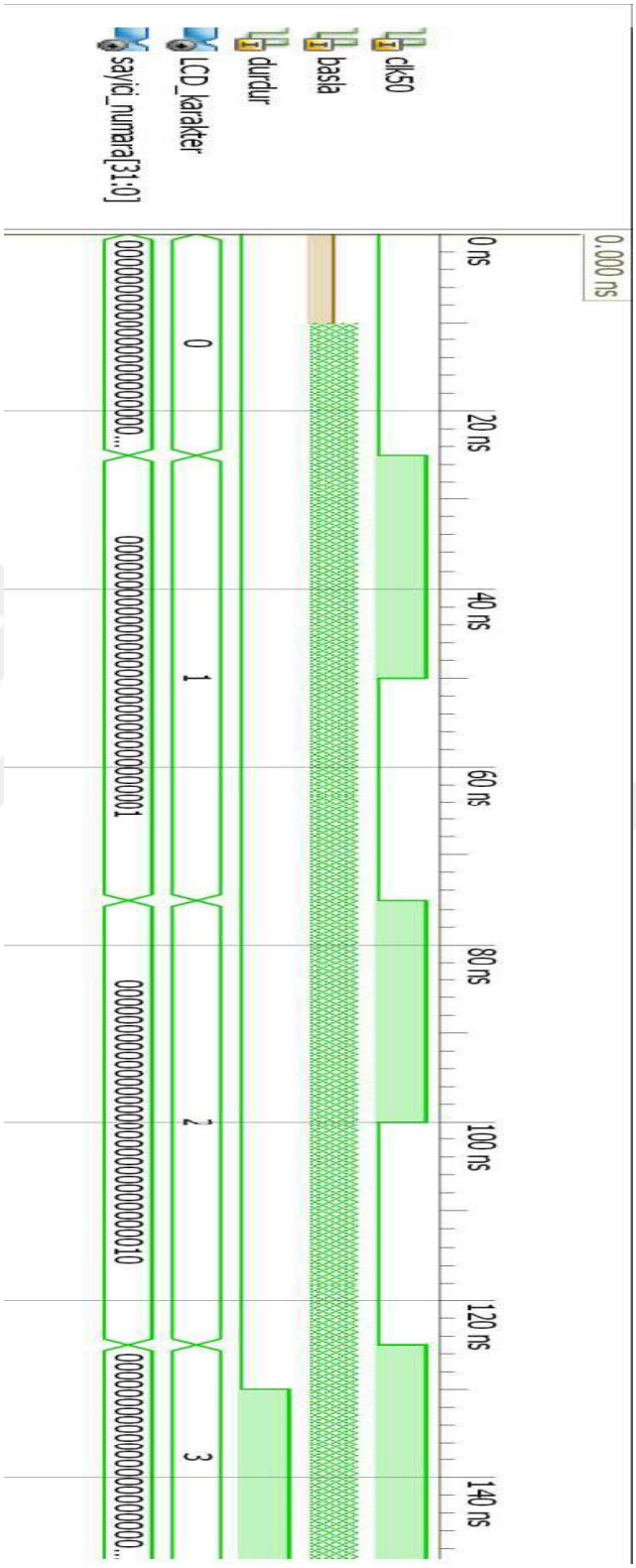
### 6.2.1 20 MHz İçin Simülasyon Sonuçları

Şekil 21'de, 20MHZ darbe üreteci için %25 dolu - %75 boş oranında seçilen darbe işaretinin simülasyon sonuçları verilmiştir.



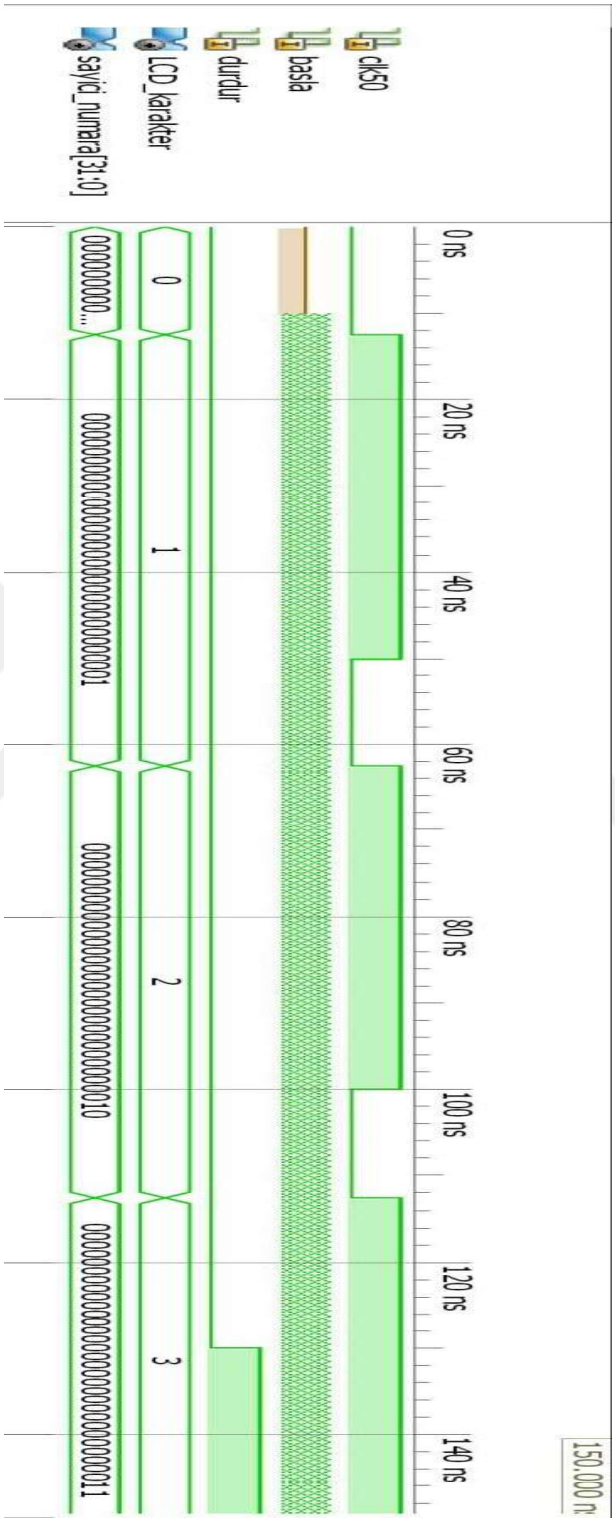

**Şekil 21:** 20 MHz Darbe İşareti %25 Dolu.

Şekil 22’de, 20MHZ darbe üreteci için %50 dolu - %50 boş oranında seçilen darbe işaretinin simülasyon sonuçları verilmiştir.




**Şekil 22:** 20 MHz Darbe İşareti %50 Dolu.

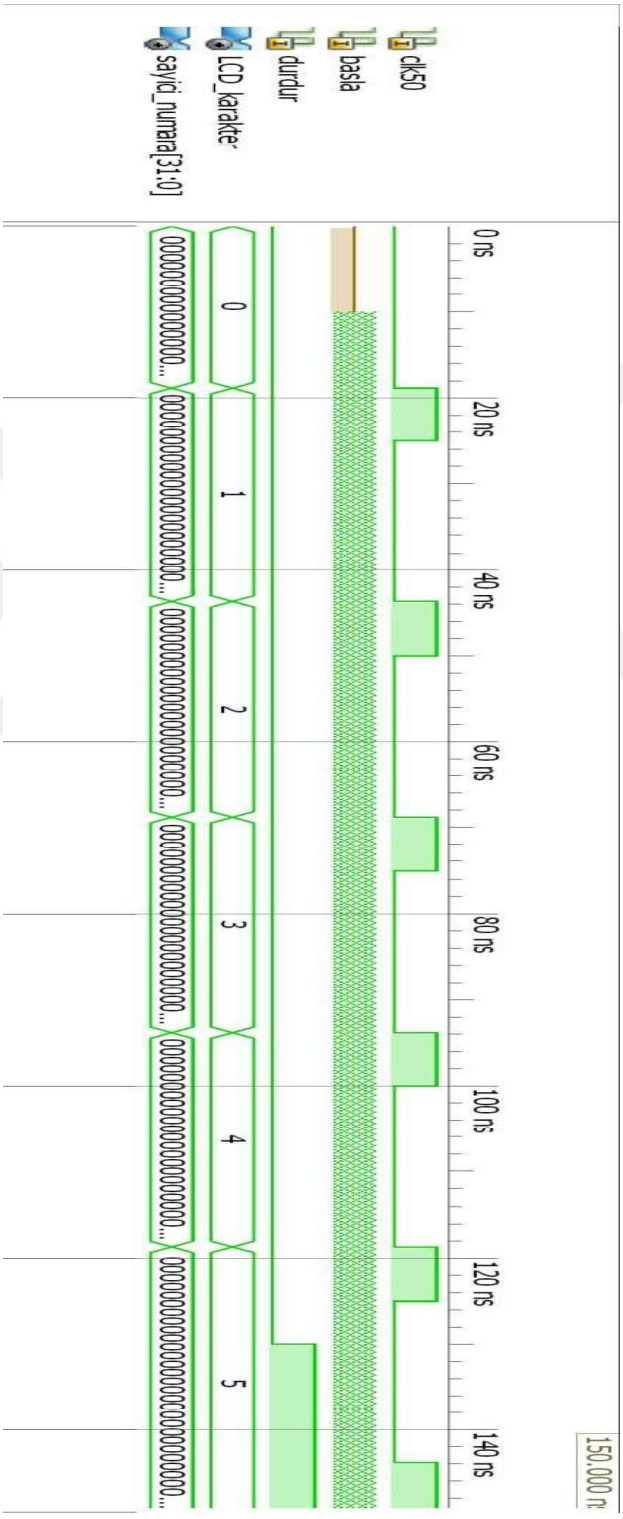
Şekil 23'te, 20MHZ darbe üreteci için %75 dolu - %25 boş oranında seçilen darbe işaretinin simülasyon sonuçları verilmiştir.




**Şekil 23:** 20 MHz Darbe İşareti %75 Dolu.

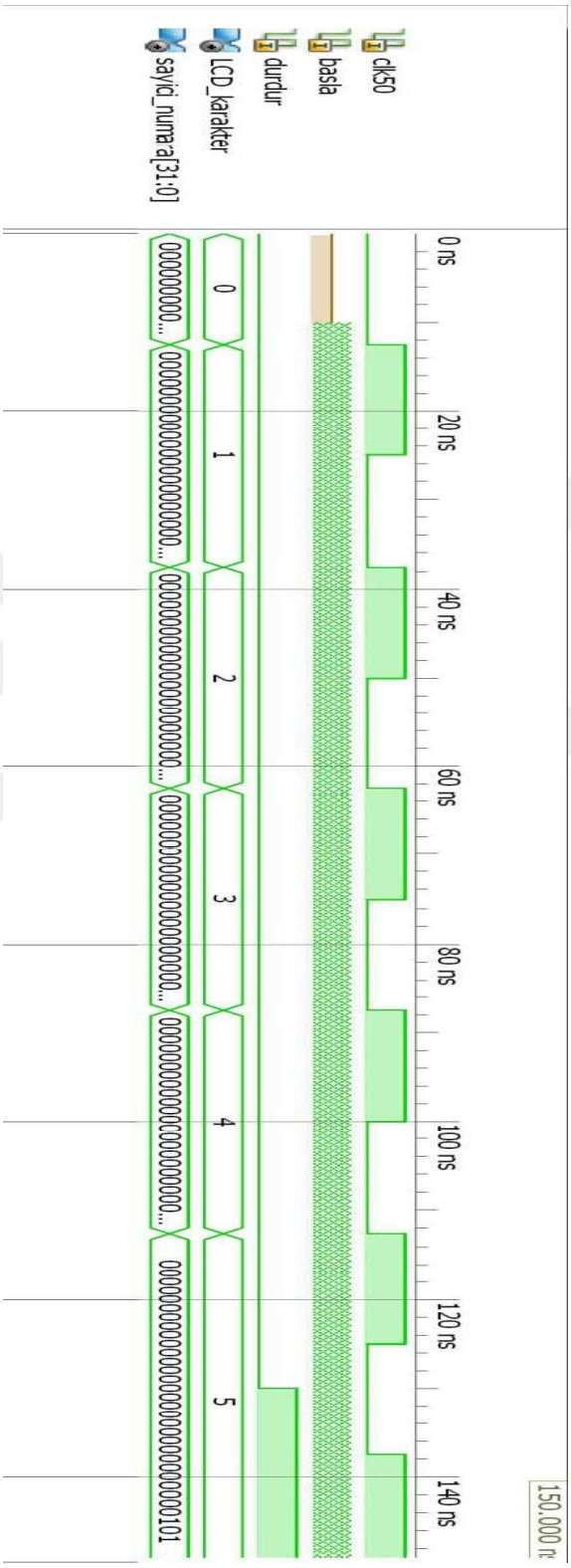
### 6.2.2 40 MHz İçin Simülasyon Sonuçları

Şekil 24'te, 40MHZ darbe üreteci için %25 dolu - %75 boş oranında seçilen darbe işaretinin simülasyon sonuçları verilmiştir.



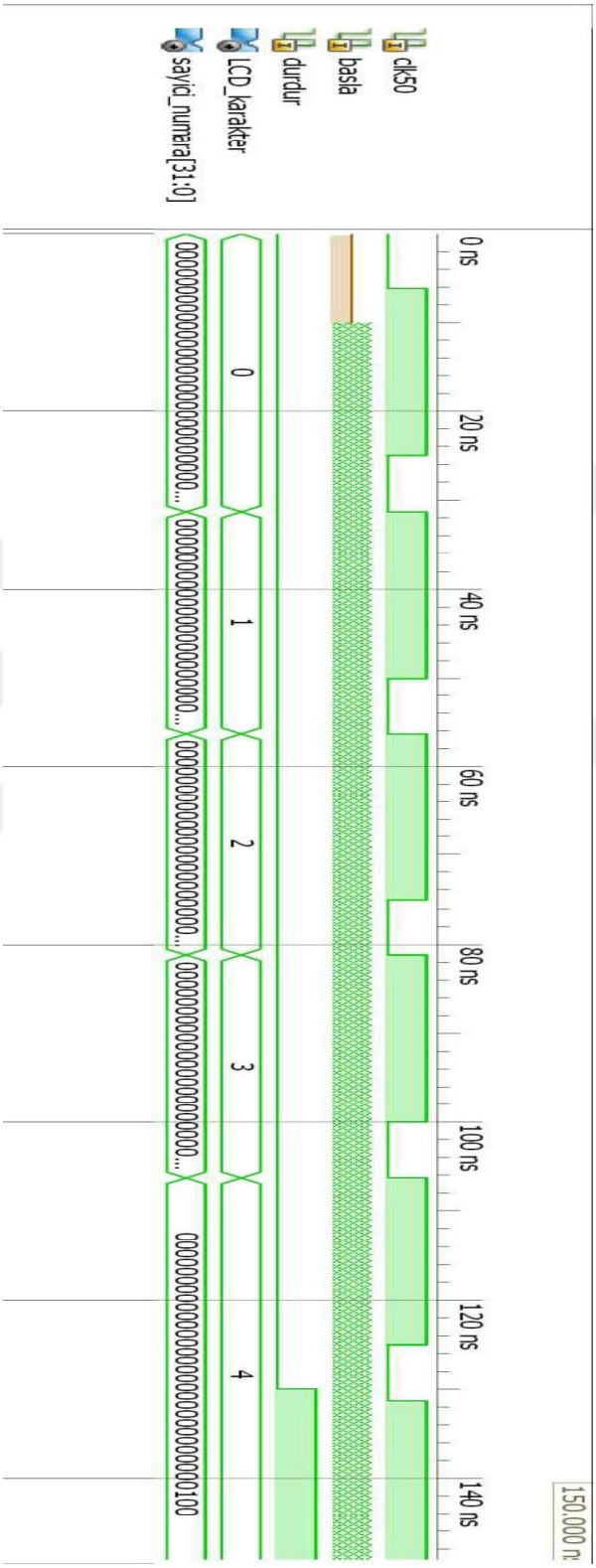

**Şekil 24:**40 MHz Darbe İşareti %25 Dolu.

Şekil 25’te, 40MHZ darbe üreteci için %50 dolu - %50 boş oranında seçilen darbe işaretinin simülasyon sonuçları verilmiştir.




**Şekil 25:** 40 MHz Darbe İşareti %50 Dolu.

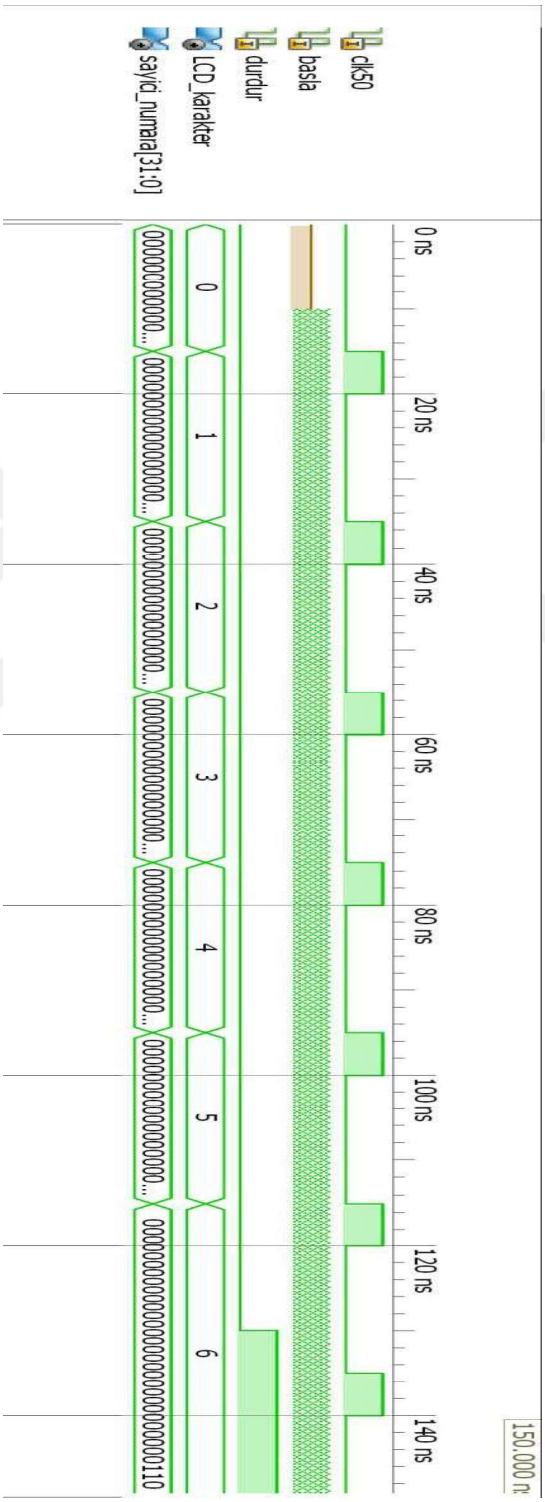
Şekil 26’da, 40MHZ darbe üreteci için %75 dolu - %25 boş oranında seçilen darbe işaretinin simülasyon sonuçları verilmiştir.




**Şekil 26:** 40 MHz Darbe İşareti %75 Dolu.

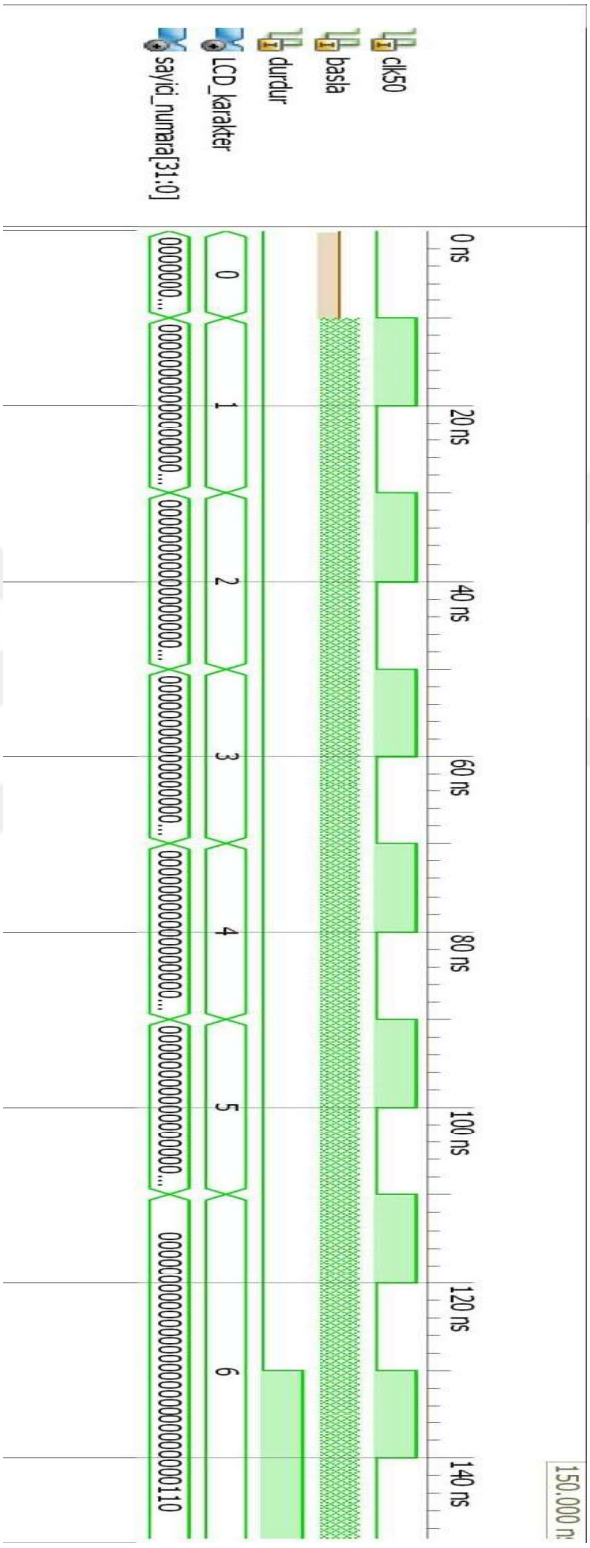
### 6.2.3 50 MHz İçin Simülasyon Sonuçları

Şekil 27'de, 50MHZ darbe üreteci için %25 dolu - %75 boş oranında seçilen darbe işaretinin simülasyon sonuçları verilmiştir.



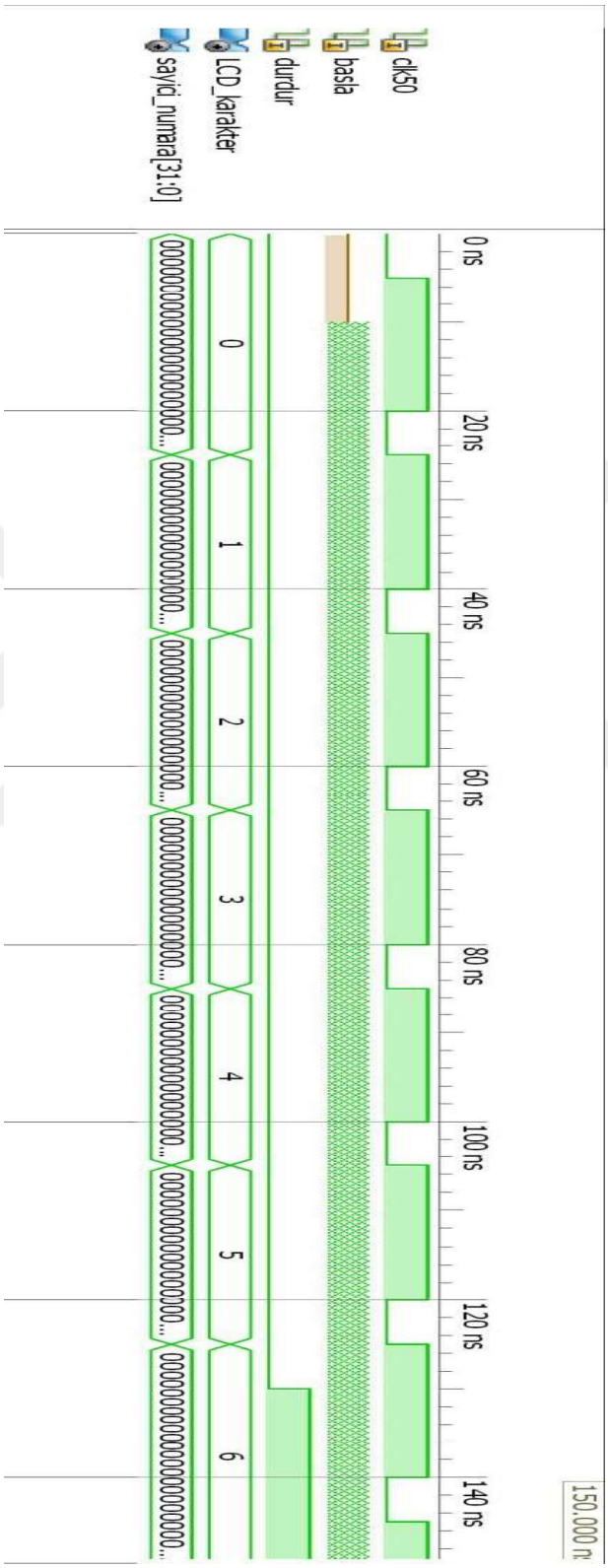

**Şekil 27:** 50 MHz Darbe İşareti %25 Dolu.

Şekil 28’de, 50MHZ darbe üreteci için %50 dolu - %50 boş oranında seçilen darbe işaretinin simülasyon sonuçları verilmiştir.




**Şekil 28:** 50 MHz Darbe İşareti %50 Dolu.

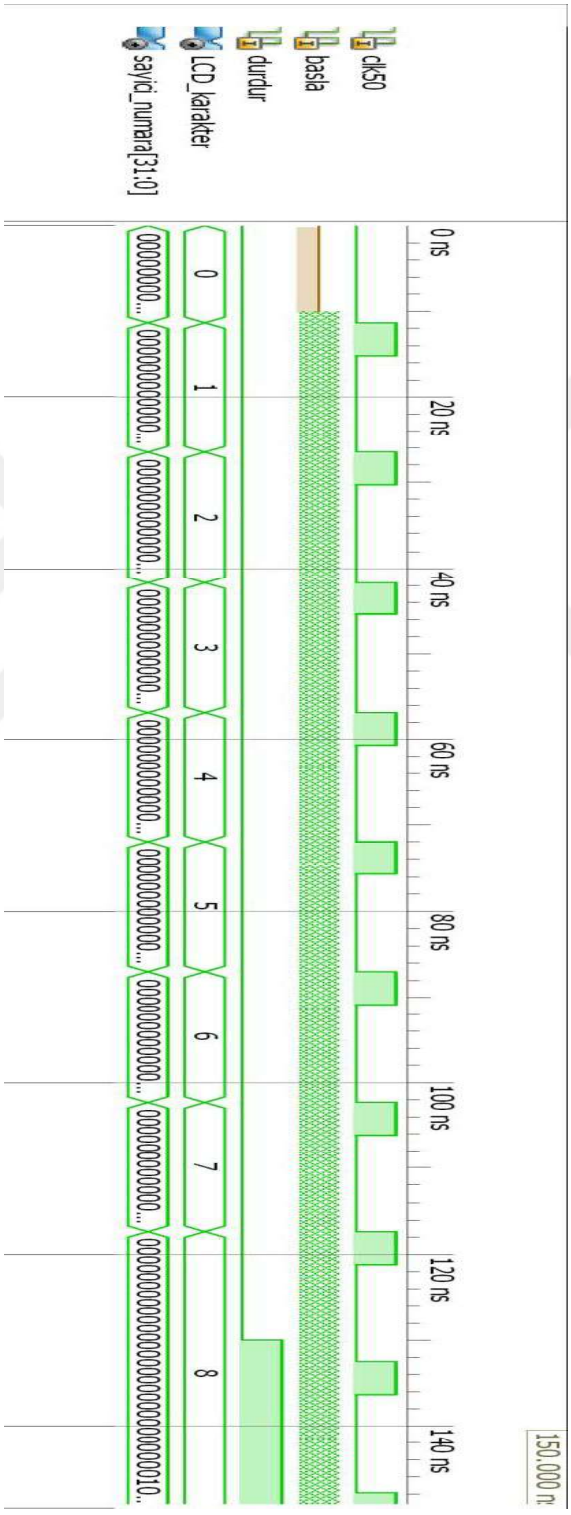
Şekil 29’da, 50MHZ darbe üreteci için %75 dolu - %25 boş oranında seçilen darbe işaretinin simülasyon sonuçları verilmiştir.




**Şekil 29:** 50 MHz Darbe İşareti %75 Dolu.

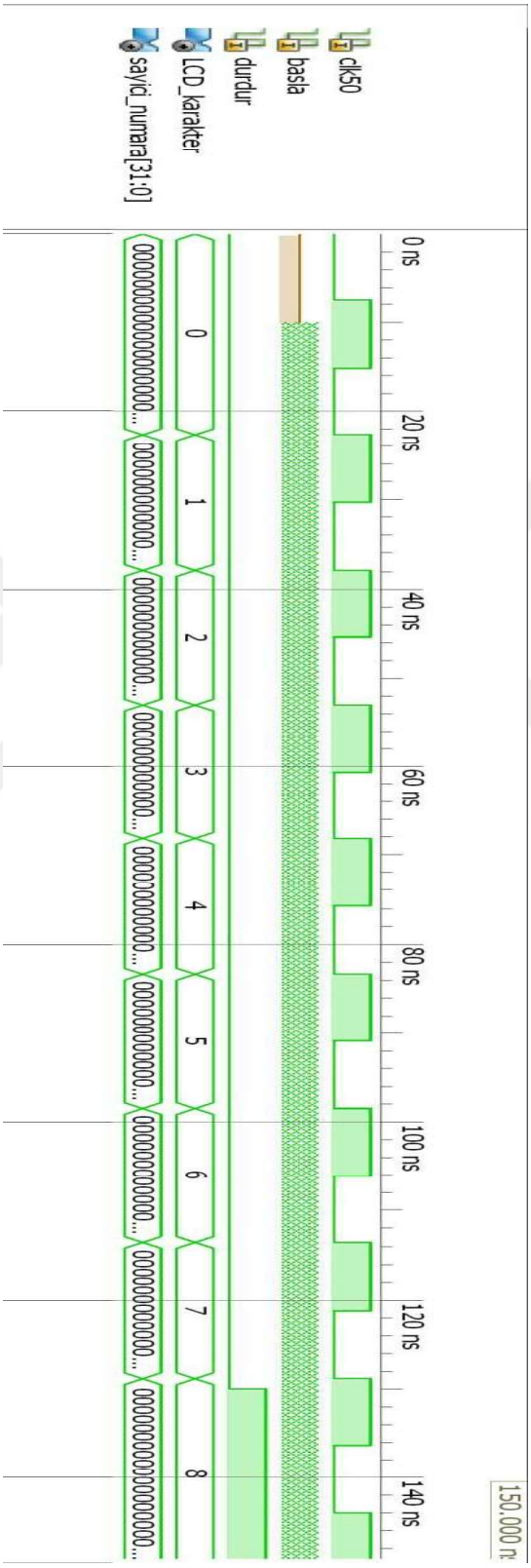
### 6.2.4 66 MHz İçin Simülasyon Sonuçları

Şekil 30'da, 66MHZ darbe üreteci için %25 dolu - %75 boş oranında seçilen darbe işaretinin simülasyon sonuçları verilmiştir.



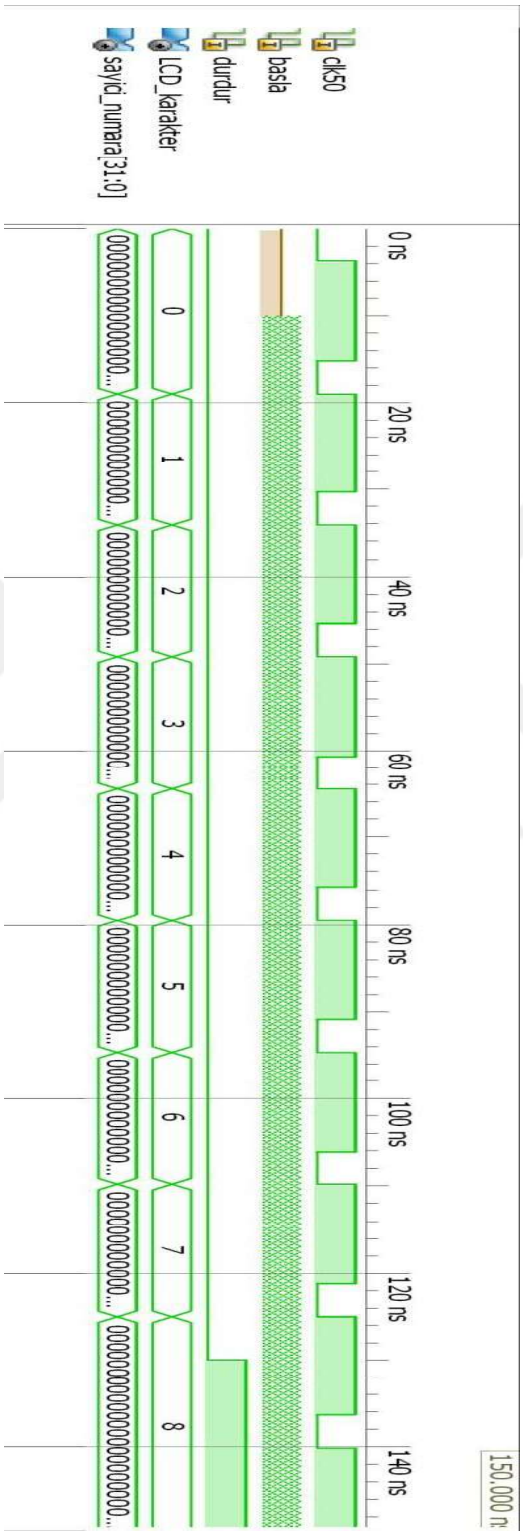

**Şekil 30:**66 MHz Darbe İşareti %25 Dolu.

Şekil 31’de, 66MHZ darbe üreteci için %50 dolu - %50 boş oranında seçilen darbe işaretinin simülasyon sonuçları verilmiştir.




**Şekil 31:** 66 MHz Darbe İşareti %50 Dolu.

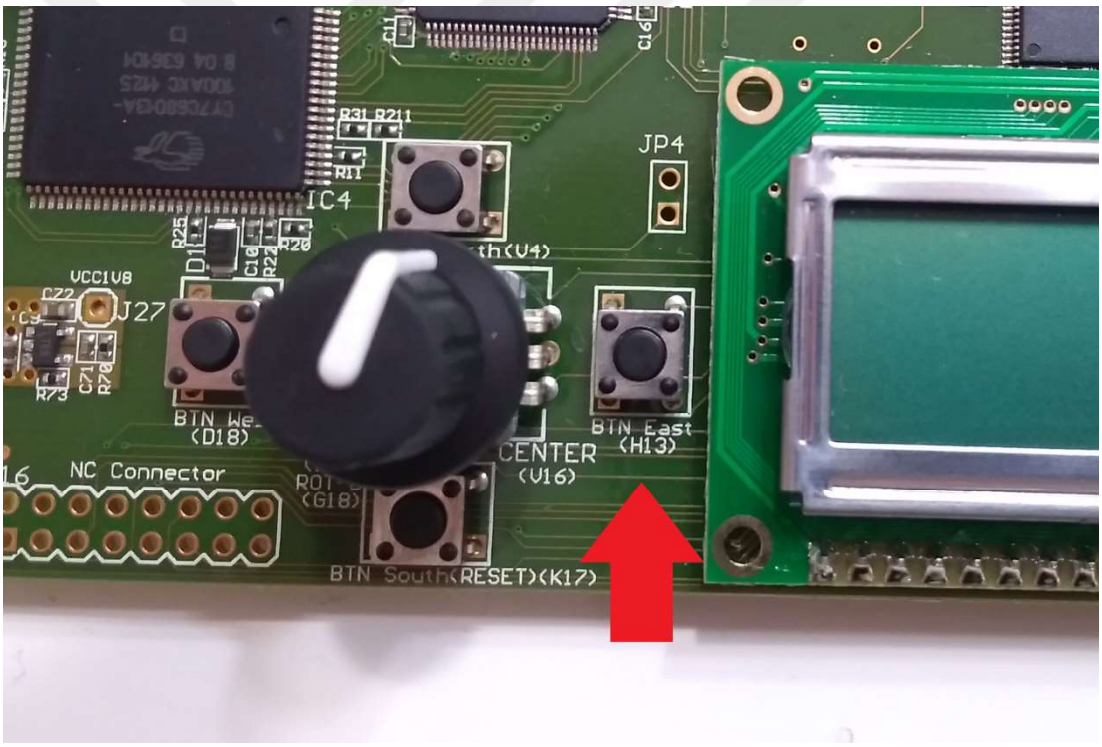
Şekil 32’de, 66MHZ darbe üreteci için %75 dolu - %25 boş oranında seçilen darbe işaretinin simülasyon sonuçları verilmiştir.



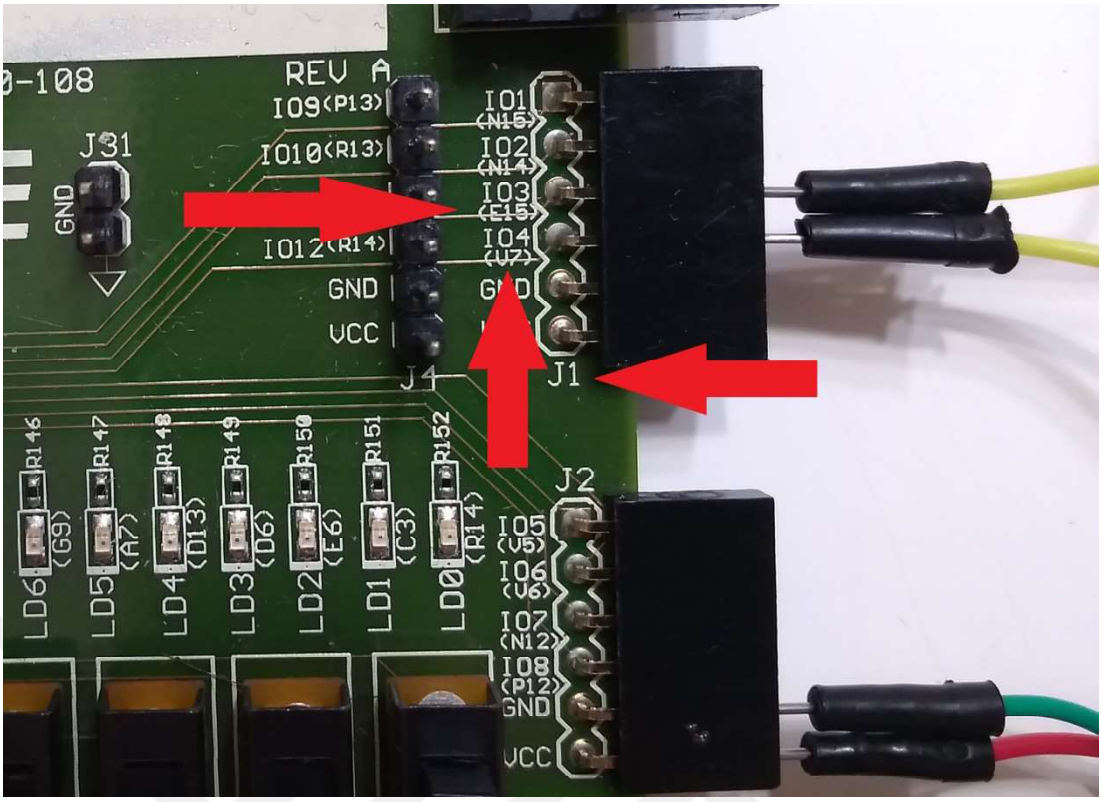

**Şekil 32:** 66 MHz Darbe İşareti %75 Dolu.

## 6.3 Test Sonuçları

Simülasyonu tamamlanan ZSD tasarımının Spartan 3E başlangıç kartında testi yapılmıştır. Başlama, durdurma ve sıfırlama giriş sinyalleri bulunmaktadır. Sıfırlama giriş sinyali H13 rumuzlu şekil 33'te belirtilen buton üzerinden uygulanmıştır. Sistemi en baştan başlatmak için kullanılmaktadır. Başlama giriş sinyali J1 rumuzlu 6'lı grup pimlerinden V7 rumuzlu şekil 34'te belirtilen pim üzerinden uygulanmıştır. ZSD bu sinyali aldığında çalışmaya başlamaktadır. Durdurma sinyali J1 rumuzlu 6'lı grup pimlerinden E15 rumuzlu şekil 34'te belirtilen pim üzerinden uygulanmıştır. ZSD bu sinyali aldığın çalışmayı durdurur. ZSD durdurma sinyali gelmediği sürece çıkışını arttırmaya devam edecektir. ZSD çıkış işareti ise LCD ekrandan verilmiştir.



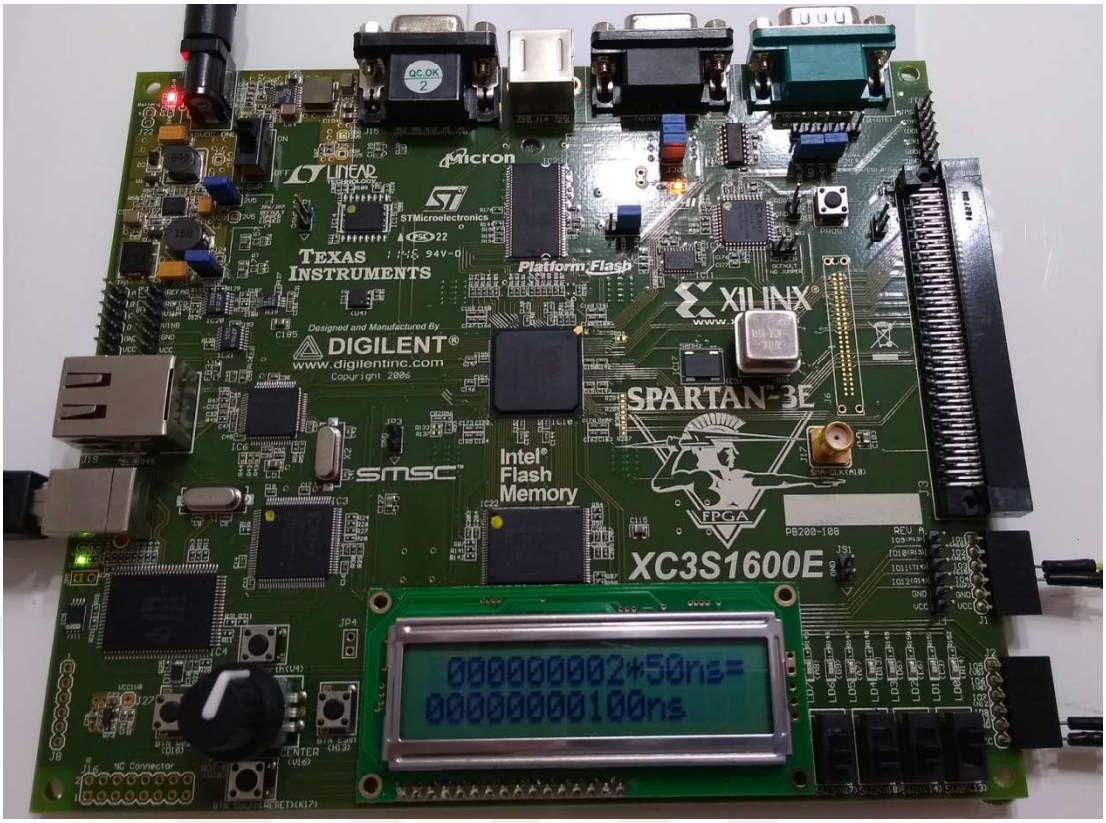

**Şekil 33:** Sıfırlama giriş butonu.




**Şekil 34:** Giriş pim grubu.

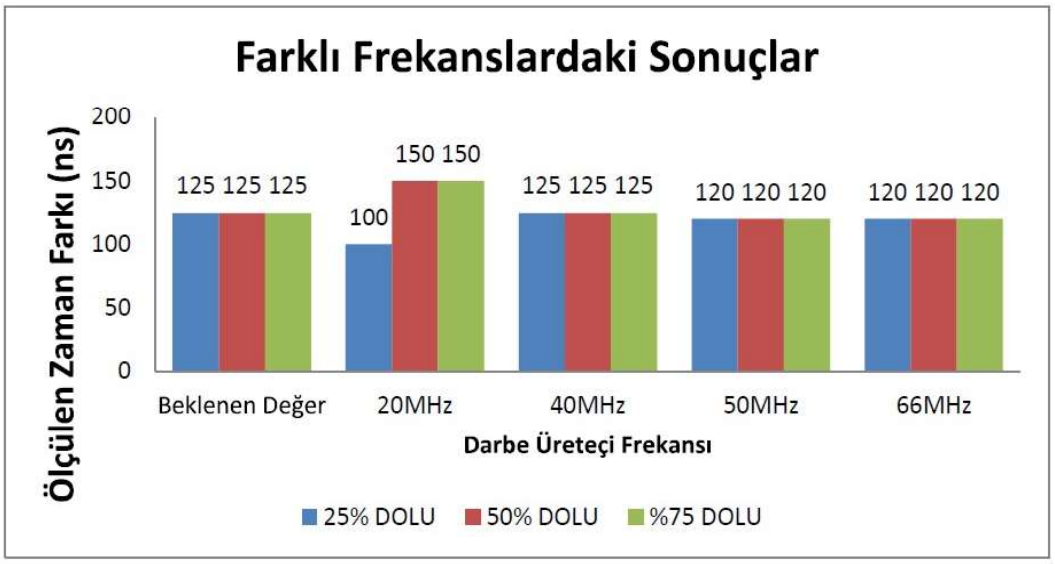
Spartan 3E başlangıç kartı için 4 farklı darbe üretecinin her birinin ürettiği darbe işareti için 3 farklı doluluk - boşluk oranları seçilip test edilmiştir. Bu doluluk - boşluk oranları %25-%75, %50-%50, %75-%25 olarak seçilmiştir. Spartan 3E kartı toplamda 12 farklı teste sokulmuştur.

Şekil 35'de, 20 MHz'lik darbe üretecinin doluluk - boşluk oranı %25 dolu - %75 boş olarak ayarlanmıştır. Bu durumdayken LCD ekran çıkış karakterleri gözlemlenmiştir.

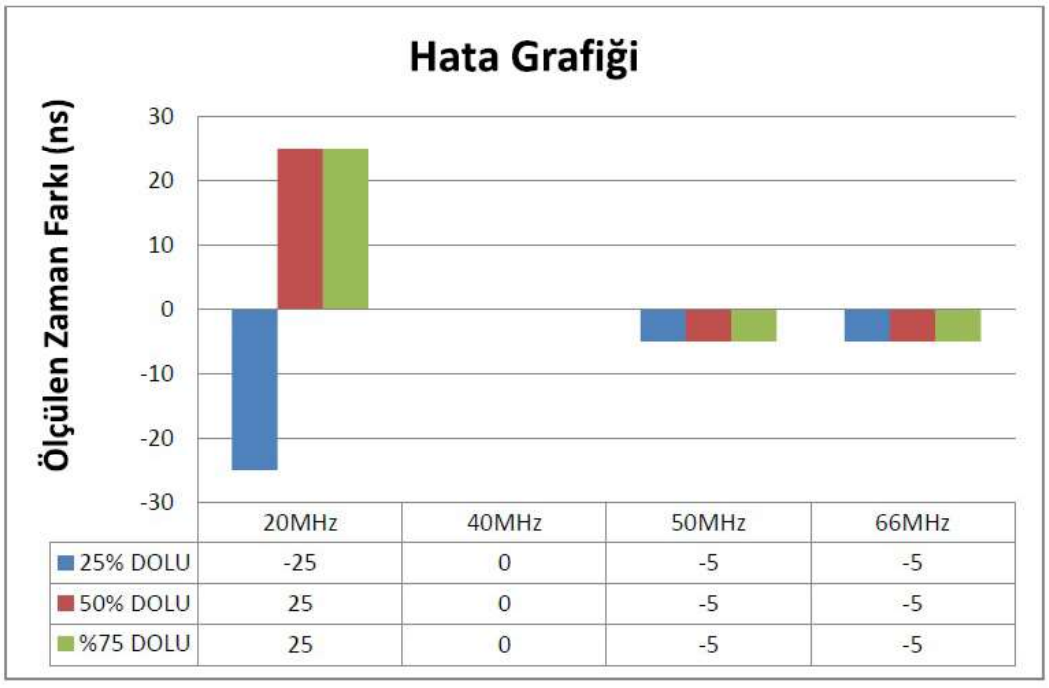



**Şekil 35:** 20 MHz darbe işareti %25 dolu.

Tasarlanan sistem kullanılan darbe üreteçlerinin frekansları 20 MHz, 40 MHz, 50 MHz ve 66 MHz'dir. Bu üreteçlerin çözünürlükleri sırasıyla 50 ns, 25 ns, 20 ns ve 15 ns'dir. Tasarlanan sisteme, 125 ns'lik bir zaman farkı oluşturularak başla işareti ile durdur işareti verilmiştir. Şekil 36'da kullanılan farklı darbe üreteçlerinin ve farklı doluluk boşluk oranlarında nasıl tepki verdiği grafik olarak gösterilmiştir. Buna karşılık olarak şekil 37'de hata grafiği verilmiştir.




**Şekil 36:** Darbe üreteçleri sonuçları.




| | 20MHz | 40MHz | 50MHz | 66MHz |
|---|---|---|---|---|
| 25% DOLU | -25 | 0 | -5 | -5 |
| 50% DOLU | 25 | 0 | -5 | -5 |
| %75 DOLU | 25 | 0 | -5 | -5 |

**Şekil 37:** Hata Grafiği.

# 7. SONUÇ

Yapılan bu çalışmada Spartan 3E başlangıç kartı XC3S1600E entegresi kullanılarak sayıcı tabanlı ZSD tasarlanmıştır. Bu tasarım4 farklı sistem çalışma frekansında ve bu çalışma frekanslarında 3 farklı doluluk – boşluk oranı seçilerek toplamda 12 farklı test yapılmıştır. Simülasyon ortamı ve gerçek ortamda sonuçların birbirlerinden farkını gözlemlemek için başla sinyali ile durdur sinyali arasında 125 ns'lik sabit bir fark seçilmiştir. Testlerin her biri başarılı bir şekilde gerçekleşmiştir.

40 MHz'lik darbe üretecinin darbe işaretinin %75 dolu olan sonucu hariç tüm test sonuçları simülasyon ortamı ve gerçek ortamda aynı çıkmıştır. 40MHz'lik darbe üretecinin darbe işaretinin %75 dolu olan simülasyonunun sonucuna göre çıkış 100 ns ölçülmüştür. Gerçek ortamdaki sonuca göre sonuç 125 ölçülmüştür. Bunun sebebi "clk50" sinyali yani sistem darbe üretecinin sinyali ile başla sinyalinin hangisinin önce başlaması gerektiğindendir.

Yazılan VHDL kodu önce "clk50" sinyalini kontrol etmektedir. Eğer "clk50" sinyali yükselen kenar veya lojik 1 durumundaysa ilk önce sıfırlama sinyali kontrol edilmektedir. Eğer sıfırlama sinyali yoksa durdurma sinyalinin olup olmadığı kontrol edilmektedir. Ardından başla sinyali kontrol edilmektedir. Bu kontrol çok hızlı bir şekilde art arda gerçekleştirmektedir. Eğer hiç biri "clk50" sinyalinin yükselen kenar veya lojik 1 durumunda aktif değilse yazılan kodun tekrar işlem yapması için "clk50" sinyalinin bir sonraki yükselen kenar veya lojik 1 durumuna gelmesi beklenmektedir. Buda bize tasarımda sistem darbe işaretinin bir periyodu kadar bir hata oluşabileceğini göstermektedir.

Tasarlanan ZSD 32bit çözünürlüğe sahiptir. Çözünürlük arttıkça hassas ölçüm aralığı artıp, kaba ölçüm aralığı azalmaktadır. Yapılan testlerde ölçüm aralığının en fazla olduğu (kaba ölçüm)darbe üreteci frekansı 20 MHz, ölçüm aralığının en az olduğu (hassas ölçüm)darbe üreteci frekansı66 MHz'dir. Bunun sebebi sistem darbe üretecinin frekansının artmasıyla çözünürlüğün doğru oranda artmasıdır.

Tasarlanan sistemde yapılan testler Spartan 3E başlangıç kartının izin verdiği ölçüde gerçekleşmiştir. Darbe üreteçlerinin ölçüm aralıkları sırasıyla aşağıda verilmiştir;

- 20 MHz'lik darbe üreteci kullanıldığında sisteme uygulanabilecek en az başlangıç süresi 50 ns, bitiş süresi ise en fazla 214 sn,
- 40 MHz'lik darbe üreteci kullanıldığında sisteme uygulanabilecek en az başlangıç süresi 25 ns, bitiş süresi ise en fazla 107 sn,
- 50 MHz'lik darbe üreteci kullanıldığında sisteme uygulanabilecek en az başlangıç süresi 20 ns, bitiş süresi ise en fazla 85 sn,
- 66 MHz'lik darbe üreteci kullanıldığında sisteme uygulanabilecek en az başlangıç süresi 15 ns, bitiş süresi ise en fazla 65 sn olduğu gözlemlenmiştir.

Sonuç olarak tasarlanan ZSD'nin, çözünürlüğünün darbe üreteci frekansının büyüklüğüne ve sayıcı bloğunun bit sayısına bağlı olduğu gözlemlenmiştir. Daha yüksek çözünürlük elde edebilmek için sistemde darbe işareti üreten darbe üreteci frekansının daha büyük, daha geniş ölçüm aralığı elde edebilmek için ise sayıcı bloğunun bit sayısının daha yüksek seçilmesi gerekmektedir.

# 8. KAYNAKLAR


Ahmed RN, Bhattacharyya S, Purkayastha BB, Bhattacharyya K (2016) "Low Resource FPGA Based Time-to-Digital Converter", ADBU-Journal of Engineering Technology ,AJET, ISSN: 2348-7305, Volume 4(1), 108 .

Aloisio A, Branchini P, Cicalese R, Giordano R, Izzo V, Loffredo S, Lomoro R (2008) "High-Resolution Time-to-Digital Converter in Field Programmable Gate Array" , DOI:10.5170/CERN-2008-008.383.

Aloisio A, Branchini P, Giordano R, Izzo V, Loffredo S (2009) "High-precision Time-to-Digital Converter in a FPGA device" , 16th IEEE-NPSS Real Time Conference, FESPP-15.

Balla A, Beretta M, Ciambrone P, Gatta M, Gonnella F, Iafolla L, Mascolo M, Messi R, Moricciani D, Riondino D (2012) "Low resource FPGA-based Time to Digital Converter", arXiv, arXiv:1206.0679 [physics.ins-det].

Büchele M, Fischer H, Herrmann F, Schaffer C (2016) "The ARAGORN Front-End - FPGA Based Implementation of a Time-to-Digital Converter" , IEEE Nuclear Science Symposium, Medical Imaging Conference and Room-Temperature Semiconductor Detector Workshop (NSS/MIC/RTSD), DOI: 10.1109/NSSMIC.2016.8069886.

Dadouche F, Turko T, Uhring W, Malass I, Bartringer J, Le Normand JP (2015) "Design Methodology of TDC on Low Cost FPGA Targets" , SENSORCOMM, The Ninth International Conference on Sensor Technologies and Applications, Volume: pp. 29-34.

Digikey, (2018), Code Download, www.digikey.com/eewiki/pages/viewpage.action?pageId=4096079#CharacterLCDModuleController(VHDL)-CodeDownload, 12 Şubat 2019.

Docplayer, HDL, docplayer.biz.tr/191549-Hardware-description-language-hdl.html, 25 Nisan 2019.

Dudek P, Szczepan´ski S, Hatfield JV (2000) "A High-Resolution CMOS Time-to-Digital Converter Utilizing a Vernier Delay Line" , IEEE Transactions On Solid-State Circuits, Vol. 35, No. 2.

FPGA, Ana Sayfa, FPGA, VHDL Ve VHDL Sözlüğü, www.fpganedir.com, 14 Nisan 2019.

GÜNEŞ EO, ÖRS SB, VHDL, web.itu.edu.tr/~ayhant/dersler/sstu/vhdl/VHDL_sunumI.pdf, 25 Nisan 2019.

Henzler S (2010) "Time-to-Digital Converters", Springer Series in Advanced Microelectronics 29, DOI:10.1007/978-90-481-8628-0 2,Springer Science+Business Media B.V.

Iucoders, VHDL donanım tanımlama dili, www.iucoders.com/attachments/VHDL.pdf, 25 Nisan 2019.

Jovanovi´c GS, Stoj^cev MK (2009) "Vernier's Delay Line Time-to-Digital Converter" Scientific Publications Of The State University Of Novi Pazar Ser. A: APPL. Math. Inform. And mech. vol. 1, 11-20.

Kara A, ASIC ve FPGA Farkları Nelerdir?, www.elektrikport.com/teknik-kutuphane/asic-ve-fpga-farklari-nelerdir/20070#ad-image-0, 7 Mayıs 2019.

Karaca Ö, Sayıcılar, kisi.deu.edu.tr//ozlem.karaca/sys9.pdf, Dokuz Eylül Üniversitesi, 19 Nisan 2019.

Keränen P (2016) "High Precision Time-To Digital Converters For Applications Requiring A Wide Measurement Range" , Doktara tezi, Oulu Üniversitesi Enstitü Üniversitesi, Bilgi Teknolojisi ve Elektrik Mühendisliği Fakültesi, Elektrik Mühendisliği, Oulu/Finlandiya.

Mahalaxmi D, Chaturvedi S, Srividya P, Narayan N, Sharma KP (2016) "Time to Digital Converter (TDC) Implementation on Spartan 3e FPGA Using

VHDL for TMTC Subsystems" , International Journal of Engineering & Technology Research Volume 4, Issue 3, May-June, pp. 33-38.

Mutukuda O (2009) Area Efficient FPGA Architecture For Datapath Circuits www.slideshare.net/omutukuda/presentation-1993175 , 14 Nisan 2019.

Rezvanyvardom M, Nejad TG, Farshidi E (2014) "A 5-bit time to digital converter using time to voltage conversion and integrating techniques for agricultural products analysis by Raman spectroscopy" , Information Processing in Agriculture, Volume 1, Issue 2, December , Pages 124-130.

Sarıtaş E, Karataş S (2015) "Her Yönüyle FPGA ve VHDL" 3.baskı, Palme Yayınevi.

Soni Z, Patel AS, Sarbadhikari AB (2017) "Comparative Study of Delay Line Based Time to Digital Converter Using FPGA" , International Research Journal of Engineering and Technology (IRJET) volume : 04 Issue : 09.

Special minds, anolog, www.nedir.com/analog, 4 mayıs 2019.

Sui T, Zhao Z, Xie S, Xie Y, Zhao Y, Huang Q, Xu J, Peng Q (2018) "A 2.3-ps RMS Resolution Time-to-Digital Converter Implemented in a Low-Cost Cyclone V FPGA", IEEE Transactions on Instrumentation and Measurement, Page(s): 1 - 14, DOI: 10.1109/TIM.2018.2880940.

T.C. MEB, Sayıcılar, immibbilisim.com/moduller/7-%20Sayıcılar.pdf, 19 Nisan 2019.

Van den Broek JDA (2012) "Design and implementation of an Analog-to-Time-to-Digital converter", Yüksek Lisans Tezi, Twente Üniversitesi, Entegre Devre Tasarımı Bölümü, Elektrik Mühendisliği, Matematik ve Bilgisayar Bilimleri Fakültesi, Enschede/Hollanda.

Wang Y, Cao Q, Liu C (2017) "A Multi-chain Merged Tapped Delay Line for High Precision Time-to-Digital Converters in FPGAs" , IEEE, Transactions on Circuits and Systems II: Express Briefs, DOI 10.1109/TCSII.2017.2698479.

Won JY, Lee JS (2016) "Time-to-Digital Converter Using a Tuned-Delay Line Evaluated in 28-, 40-, and 45-nm FPGAs", IEEE Transactions On Instrumentation And Measurement, VOL. 65, NO. 7.

Xilinx (2007), MicroBlaze Development Kit Spartan-3E 1600E Edition User Guide, www.xilinx.com/support/documentation/boards_and_kits/ug257.pdf, 25 Nisan 2019.

Xilinx (2018), Introduction, www.xilinx.com/support/documentation/data_sheets/ds312.pdf, 7 Mayıs 2019.

Xing N, Woo JK, Shin WY, Lee H, Kim S (2010) "A 14.6 ps Resolution, 50 ns Input-Range Cyclic Time-to-Digital Converter Using Fractional Difference Conversion Method" , IEEE Transactions On Circuits And Systems-I: Regular Papers, Vol. 57, No. 12, December.

Yu J, Dai FF, Jaeger RC (2010) "A 12-Bit Vernier Ring Time-to-Digital Converter in 0.13 µm CMOS Technology" , IEEE Journal Of Solid-State Circuits, Vol. 45, No. 4.

Zhang M, Jiang J, Wang Q, Huang J (2018) "A Self-Timed Ring Based Time-to-Digital Converter on FPGA" , Conference: 14th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT).

# EKLER

# 9. EKLER

### EK A.1 - Zaman/Sayısal Dönüştürücü Ana Kodu

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity TDC is
port(clk50, basla, durdur, reset : in STD_LOGIC;
LCD_CIKIS : out STD_LOGIC_VECTOR(10 downto 0));
end TDC;

architecture Behavioral of TDC is

component SAYICI -- SAYICI TASARIMI ON TANIMLAMA
port (clk,basla,reset,durdur : in STD_LOGIC;
CIKIS : out STD_LOGIC_VECTOR(31 downto 0));
end component;

-- LCD KOD COZUCU ON TANIMLAMA
component LCD_KOD_COZUCU
port(vector_in : in STD_LOGIC_VECTOR(31 downto 0);
UST_SATIR, ALT_SATIR : out STD_LOGIC_VECTOR(127 downto 0));
end component;

-- LCD SURME ON TANIMLARI
component lcd_controller
port(clk        : IN    STD_LOGIC;  --SİSTEM DARBE İŞARETİ
reset_n    : IN    STD_LOGIC;
rw, rs, e  : OUT   STD_LOGIC;  --YAZMA,OKUMA VE İŞLEM
lcd_data   : OUT   STD_LOGIC_VECTOR(7 DOWNTO 0); --VERİ SİNYALİ
line1_buffer : IN STD_LOGIC_VECTOR(127 downto 0); -- ÜST SATIR LCD
line2_buffer : IN STD_LOGIC_VECTOR(127 downto 0)); -- ALT SATIR LCD
end component;

--  LCD ICIN BAZI SINYALLER URETILMISTIR.
signal sayici_numara : STD_LOGIC_VECTOR(31 downto 0);
signal SATIR_1,SATIR_2 : STD_LOGIC_VECTOR(127 downto 0);
signal LCD_temp : STD_LOGIC_VECTOR(10 downto 0); -- 7:0 --> lcd data
                                                 -- 8 rw / 9 rs / 10 e
```

**EK A.1**

```
begin

SAY : SAYICI
port map(clk => clk50, basla => basla, reset => reset, durdur => durdur, CIKIS
=> sayici_numara);

LCDKOD : LCD_KOD_COZUCU



port map(vector_in => sayici_numara, UST_SATIR => SATIR_1 , ALT_SATIR
=> SATIR_2);

DRIVER : lcd_controller
port map(clk => clk50, reset_n => not(reset), rw => LCD_temp(8), rs =>
LCD_temp(9), e => LCD_temp(10), lcd_data => LCD_temp(7 downto 0),
line1_buffer => SATIR_1, line2_buffer => SATIR_2);

LCD_CIKIS <= LCD_temp;

end Behavioral;
```

**EK A.2 - 32 Bitlik Sayıcı Yardımcı Alt Kodu**

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_unsigned.ALL;

entity SAYICI is
port (clk,basla,reset,durdur : in STD_LOGIC;
CIKIS : out STD_LOGIC_VECTOR(31 downto 0));
end SAYICI;

architecture Behavioral of SAYICI is
signal temp : STD_LOGIC_VECTOR(31 downto 0) := (others => '0');
begin
SAYICI_process : process(clk,basla,reset,durdur) -- PARANTEZ ICINDEKILERE
-- GORE ISLEM YAP
begin
if clk'event and clk='1' then -- CLOCK ISARETIN LOJIK 1 VE YUKSELEN
--KENARINDA ISLEM YAP
if reset = '1' then -- RESET LOJIK 1 ISE SAYMAYI SIFIRLA
temp <= "00000000000000000000000000000000";
elsif durdur = '1' then -- DURDUR ISARETI LOJIK 1 SE SAYMAYI DURDUR
--    VE SON DURUMDA KAL
temp <= temp ;
elsif basla = '1' then -- BASLA ISARETI LOJIK 1 SE SAYMAYA BASLA
temp <= temp + '1';
end if;
end if;
end process;
CIKIS <= temp;
end Behavioral;
```

**EK A.3 - LCD Enkoder Ana Kodu**

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity LCD_KOD_COZUCU is
port(vector_in : in STD_LOGIC_VECTOR(31 downto 0);
UST_SATIR, ALT_SATIR : out STD_LOGIC_VECTOR(127 downto 0));
end LCD_KOD_COZUCU;

architecture Behavioral of LCD_KOD_COZUCU is

component SATIR1_ENKODER -- UST SATIR ISLEMLERI
port( giren : in STD_LOGIC_VECTOR(31 downto 0);
CIKAN : out STD_LOGIC_VECTOR(127 downto 0));
end component;

component SATIR2_ENKODER -- ALT SATIR ISLEMLERI
port( giren : in STD_LOGIC_VECTOR(31 downto 0);
CIKAN : out STD_LOGIC_VECTOR(127 downto 0));
end component;

begin
SATIR1: SATIR1_ENKODER
port map(giren => vector_in, CIKAN => UST_SATIR);

SATIR2: SATIR2_ENKODER
port map(giren => vector_in, CIKAN => ALT_SATIR);

end Behavioral
```

## EK A.4 – LCDEnkoderi Satır 1 Yardımcı Kodu

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;


entity SATIR1_ENKODER is
port(giren : in STD_LOGIC_VECTOR(31 downto 0);
CIKAN : out STD_LOGIC_VECTOR(127 downto 0));
end SATIR1_ENKODER;

architecture Behavioral of SATIR1_ENKODER is

-- BOLME ISLEMI, BUNUN AMACI 32 BITLIK SAYIYI
-- 4 BITLIK SAYILARA DONUSTURMEK
component BOLME_ISLEMI
port(input : in STD_LOGIC_VECTOR(31 downto 0);
d1,d2,d3,d4,d5,d6,d7,d8,d9 : out STD_LOGIC_VECTOR(3 downto 0));
end component;


-- LCD YE VERI DONUSTURME VEYA AKTARMA ISLEMI
component DONUSUM1
port( input0, input1, input2, input3, input4, input5, input6, input7 , input8 : in
STD_LOGIC_VECTOR(7 downto 0);
output : out STD_LOGIC_VECTOR(127 downto 0));
end component;

-- KOD COZME ISLEMI, KARAKTER LCD DATASHET'NE GORE
component KOD_COZUCU
port(input : in STD_LOGIC_VECTOR(3 downto 0);
output : out STD_LOGIC_VECTOR(7 downto 0));
end component;

--
signal temp0,temp1,temp2,temp3,temp4,temp5,temp6,temp7,temp8 :
STD_LOGIC_VECTOR(3 downto 0);

signal u_temp0,u_temp1,u_temp2,u_temp3,u_temp4,u_temp5,u_temp6,
u_temp7,u_temp8 : STD_LOGIC_VECTOR(7 downto 0);

begin
BOL : BOLME_ISLEMI
port map(input => giren ,d1 => temp0, d2 => temp1, d3 => temp2, d4 => temp3,
d5 => temp4, d6 => temp5, d7 => temp6, d8 => temp7, d9 => temp8);
```

**EK A.4**

```
        EN0 : KOD_COZUCU
                port map(input => temp0, output => u_temp0);
        EN1 : KOD_COZUCU
                port map(input => temp1, output => u_temp1);
        EN2 : KOD_COZUCU
                port map(input => temp2, output => u_temp2);
        EN3 : KOD_COZUCU
                port map(input => temp3, output => u_temp3);
        EN4 : KOD_COZUCU
                port map(input => temp4, output => u_temp4);
        EN5 : KOD_COZUCU
                port map(input => temp5, output => u_temp5);
        EN6 : KOD_COZUCU
                port map(input => temp6, output => u_temp6);
        EN7 : KOD_COZUCU
                port map(input => temp7, output => u_temp7);
        EN8 : KOD_COZUCU
                port map(input => temp8, output => u_temp8);
        DONUSTUR : DONUSUM1
                port map(input0 => u_temp8, input1 => u_temp7, input2 =>
u_temp6, input3 => u_temp5, input4 => u_temp4, input5 => u_temp3, input6 =>
u_temp2, input7 => u_temp1, input8 => u_temp0, output => CIKAN);

end Behavioral;
```

**EK A.5 - LCD Enkoderi Satır 2 Yardımcı Kodu**

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;


entity SATIR2_ENKODER is
port(giren : in STD_LOGIC_VECTOR(31 downto 0);
CIKAN : out STD_LOGIC_VECTOR(127 downto 0));
end SATIR2_ENKODER;

architecture Behavioral of SATIR2_ENKODER is
-- BOLME ISLEMI, BUNUN AMACI 32 BITLIK SAYIYI
-- 4 BITLIK SAYILARA DONUSTURMEK
component BOLME_ISLEMI2
port(input : in STD_LOGIC_VECTOR(31 downto 0);
d1,d2,d3,d4,d5,d6,d7,d8,d9,d10 : out STD_LOGIC_VECTOR(3 downto 0));
end component;

-- LCD YE VERI DONUSTURME VEYA AKTARMA ISLEMI
component DONUSUM2
port( input0, input1, input2, input3, input4, input5, input6, input7 , input8 , input9
: in STD_LOGIC_VECTOR(7 downto 0);
output : out STD_LOGIC_VECTOR(127 downto 0));
end component;

-- KOD COZME ISLEMI, KARAKTER LCD DATASHET'NE GORE
component KOD_COZUCU
port(input : in STD_LOGIC_VECTOR(3 downto 0);
output : out STD_LOGIC_VECTOR(7 downto 0));
end component;

signal temp0,temp1,temp2,temp3,temp4,temp5,temp6,temp7,temp8,temp9 :
STD_LOGIC_VECTOR(3 downto 0);
signal u_temp0,u_temp1,u_temp2,u_temp3,u_temp4,u_temp5,u_temp6,u_temp7,
u_temp8,u_temp9 : STD_LOGIC_VECTOR(7 downto 0);

begin
BOL : BOLME_ISLEMI2
port map(input => giren ,d1 => temp0, d2 => temp1, d3 => temp2, d4 => temp3,
d5 => temp4, d6 => temp5, d7 => temp6, d8 => temp7, d9 => temp8, d10 =>
temp9);
        EN0 : KOD_COZUCU
                port map(input => temp0, output => u_temp0);
        EN1 : KOD_COZUCU
                port map(input => temp1, output => u_temp1);
        EN2 : KOD_COZUCU
                port map(input => temp2, output => u_temp2);
```

**EK A.5**

```
        EN3 : KOD_COZUCU
                port map(input => temp3, output => u_temp3);
        EN4 : KOD_COZUCU
                port map(input => temp4, output => u_temp4);
        EN5 : KOD_COZUCU
                port map(input => temp5, output => u_temp5);
        EN6 : KOD_COZUCU
                port map(input => temp6, output => u_temp6);
        EN7 : KOD_COZUCU
                port map(input => temp7, output => u_temp7);
        EN8 : KOD_COZUCU
                port map(input => temp8, output => u_temp8);
        EN9 : KOD_COZUCU
                port map(input => temp9, output => u_temp9);

        DONUSTUR : DONUSUM2
                port map(input0 => u_temp9, input1 => u_temp8, input2 =>
u_temp7, input3 => u_temp6, input4 => u_temp5, input5 => u_temp4, input6 =>
u_temp3, input7 => u_temp2, input8 => u_temp1, input9 => u_temp0, output =>
CIKAN);
end Behavioral;
```

**EK A.6 - Bölme İşlemı Yapan Alt Program Kodu**

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;


entity BOLME_ISLEMI is
port(input : in STD_LOGIC_VECTOR(31 downto 0);
d1,d2,d3,d4,d5,d6,d7,d8,d9 : out STD_LOGIC_VECTOR(3 downto 0));
end BOLME_ISLEMI;

architecture Behavioral of BOLME_ISLEMI is

component alt_bolme_islemi
generic(divisor :integer := 10); -- BOLEN
port(divident : in STD_LOGIC_VECTOR(31 downto 0); -- BOLUNEN SAYI
quotient : out STD_LOGIC_VECTOR(3 downto 0); -- BOLEN
remainder : out STD_LOGIC_VECTOR(31 downto 0)); -- KALAN SAYI
end component;

signal q1,q2,q3,q4,q5,q6,q7,q8 : STD_LOGIC_VECTOR(3 downto 0);
signal r1,r2,r3,r4,r5,r6,r7,r8 : STD_LOGIC_VECTOR(31 downto 0);
begin
DIV1 : alt_bolme_islemi -- GİRİSİ BOLDU, BOLUM 1 VE KALAN 1 CİKTİ
                generic map( divisor => 100000000)
                port map(divident => input, quotient => q1, remainder => r1);
DIV2 : alt_bolme_islemi -- BOLUM 1 İ BOLDU, BOLUM 2 VE KALAN 2 CİKTİ
                generic map( divisor => 10000000)
                port map(divident => r1, quotient => q2, remainder => r2);
DIV3 : alt_bolme_islemi -- BOLUM 2 İ BOLDU, BOLUM 3 VE KALAN 3 CİKTİ
                generic map( divisor => 1000000)
                port map(divident => r2, quotient => q3, remainder => r3);
DIV4 : alt_bolme_islemi -- BOLUM 3 İ BOLDU, BOLUM 4 VE KALAN 4 CİKTİ
                generic map( divisor => 100000)
                port map(divident => r3, quotient => q4, remainder => r4);
DIV5 : alt_bolme_islemi -- BOLUM 4 İ BOLDU, BOLUM 5 VE KALAN 5 CİKTİ
                generic map( divisor => 10000)
                port map(divident => r4, quotient => q5, remainder => r5);
DIV6 : alt_bolme_islemi -- BOLUM 5 İ BOLDU, BOLUM 6 VE KALAN 6 CİKTİ
                generic map( divisor => 1000)
                port map(divident => r5, quotient => q6, remainder => r6);
DIV7 : alt_bolme_islemi -- BOLUM 6 İ BOLDU, BOLUM 7 VE KALAN 7 CİKTİ
                generic map( divisor => 100)
                port map(divident => r6, quotient => q7, remainder => r7);
DIV8 : alt_bolme_islemi -- BOLUM 7 İ BOLDU, BOLUM 8 VE KALAN 8 CİKTİ
                generic map( divisor => 10)
                port map(divident => r7, quotient => q8, remainder => r8);
```

**EK A.6**

```
d1 <= q1; --KALAN 1
                    d2 <= q2; --KALAN 2
                    d3 <= q3; --KALAN3
                    d4 <= q4; -- KALAN4
                    d5 <= q5; -- KALAN5
                    d6 <= q6; -- KALAN6
                    d7 <= q7; -- KALAN7
                    d8 <= q8; -- KALAN8
                    d9 <= r8(3 downto 0); -- BOLUM 8
-- 0-9 ARASİNA GORE CIKAN SONUCLAR ELDE EDİLDİ.

end Behavioral;
```

## EK A.7 - Alt Bölme İşlemi Yapan Yardımcı Kodu

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;



entity alt_bolme_islemi is
generic(divisor : integer := 10); -- bolen
port(divident : in STD_LOGIC_VECTOR(31 downto 0);-- bolunen sayi
quotient : out STD_LOGIC_VECTOR(3 downto 0); -- bolum
remainder : out STD_LOGIC_VECTOR(31 downto 0)); -- kalan

end alt_bolme_islemi;

architecture Behavioral of alt_bolme_islemi is

signal q : STD_LOGIC_VECTOR(3 downto 0);
signal r : STD_LOGIC_VECTOR(31 downto 0);
begin
div_process : process(divident)
variable n : UNSIGNED(31 downto 0);
variable t : STD_LOGIC_VECTOR(3 downto 0);
variable z : UNSIGNED(31 downto 0);
begin
z := to_unsigned(divisor, z'length);
t := "0000";
n := unsigned(divident);
for i in 0 to 9 loop
if n >= z then
n := n - z;
t := t + '1';
end if;
end loop;
q <= t;
r <= std_logic_vector(n);
end process;
quotient <= q;
remainder <= r;

end Behavioral;
```

## EK A.8 - LCD Sürücü Kodu

(Sf.1-4)

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY lcd_controller IS
PORT(
clk        : IN    STD_LOGIC;
reset_n    : IN    STD_LOGIC;
rw, rs, e  : OUT   STD_LOGIC;
lcd_data   : OUT   STD_LOGIC_VECTOR(7 DOWNTO 0);
          line1_buffer : IN STD_LOGIC_VECTOR(127 downto 0);
          line2_buffer : IN STD_LOGIC_VECTOR(127 downto 0));
END lcd_controller;

ARCHITECTURE controller OF lcd_controller IS
TYPE CONTROL IS(power_up, initialize, RESETLINE, line1, line2, send);
SIGNAL    state      : CONTROL;
CONSTANT  freq       : INTEGER := 50; -- MHz OLARAK SİSTEM
-- FREKANSI BURAYA YAZILACAKTIR.
SIGNAL            ptr          : natural range 0 to 16 := 15;
SIGNAL            line                    : STD_LOGIC := '1';
BEGIN
PROCESS(clk)
VARIABLE clk_count : INTEGER := 0;
BEGIN
IF(clk'EVENT and clk = '1') THEN

CASE state IS

WHEN power_up =>
IF(clk_count < (50000 * freq)) THEN
clk_count := clk_count + 1;
state <= power_up;
ELSE
clk_count := 0;
rs <= '0';
rw <= '0';
lcd_data <= "00110000";
state <= initialize;
END IF;

WHEN initialize =>
clk_count := clk_count + 1;
IF(clk_count < (10 * freq)) THEN
lcd_data <= "00111100";
e <= '1';
state <= initialize;
ELSIF(clk_count < (60 * freq)) THEN
lcd_data <= "00000000";
e <= '0';
state <= initialize;
ELSIF(clk_count < (70 * freq)) THEN
lcd_data <= "00001100";

e <= '1';
state <= initialize;
ELSIF(clk_count < (120 * freq)) THEN   --50 us bekle
```

## EK A.8

```
        lcd_data <= "00000000";

e <= '0';
state <= initialize;
ELSIF(clk_count < (130 * freq)) THEN   --ekranı temizle
lcd_data <= "00000001";
e <= '1';
state <= initialize;
ELSIF(clk_count < (2130 * freq)) THEN  --2 ms bekle
lcd_data <= "00000000";
e <= '0';
state <= initialize;
ELSIF(clk_count < (2140 * freq)) THEN
lcd_data <= "00000110";
e <= '1';
state <= initialize;
ELSIF(clk_count < (2200 * freq)) THEN  -- 60 us bekle
lcd_data <= "00000000";
e <= '0';
state <= initialize;
ELSE
clk_count := 0;
state <= RESETLINE;
END IF;

WHEN resetline =>
                                ptr <= 16;
                                if line = '1' then
                        lcd_data <= "10000000";
rs <= '0';
rw <= '0';
clk_count := 0;
state <= send;
                                else
                                lcd_data <= "11000000";
rs <= '0';
rw <= '0';
clk_count := 0;
state <= send;
                                end if;
WHEN line1 =>
line <= '1';
lcd_data <= line1_buffer(ptr*8 + 7 downto ptr*8);
rs <= '1';
rw <= '0';
clk_count := 0;
line <= '1';
state <= send;

WHEN line2 =>
line <= '0';
lcd_data <= line2_buffer(ptr*8 + 7 downto ptr*8);
rs <= '1';
rw <= '0';
clk_count := 0;
state <= send;
```

## EK A.8

```
WHEN send =>

IF(clk_count < (50 * freq)) THEN
IF(clk_count < freq) THEN
        e <= '0';
ELSIF(clk_count < (14 * freq)) THEN
        e <= '1';
ELSIF(clk_count < (27 * freq)) THEN
        e <= '0';
END IF;
clk_count := clk_count + 1;
state <= send;


ELSE
clk_count := 0;
if line = '1' then
if ptr = 0 then
        line <= '0';
        state <= resetline;


else
ptr <= ptr - 1;
state <= line1;
end if;
else
if ptr = 0 then
        line <= '1';
state <= resetline;
else
ptr <= ptr - 1;
state <= line2;
end if;
end if;
END IF;
END CASE;

IF(reset_n = '0') THEN
state <= power_up;
END IF;

END IF;
END PROCESS;
END controller;
```

--[33].

## EK A.9 - SPARTAN 3E Pim Tanımlamaları

```
NET "basla" LOC = "L13" | IOSTANDARD = LVTTL | PULLUP ;
NET "durdur" LOC = "L14" | IOSTANDARD = LVTTL | PULLUP ;

NET "CLK50" LOC = "C9" | IOSTANDARD = LVCMOS33 ;
NET "reset" LOC = "H13" | IOSTANDARD = LVTTL | PULLDOWN ;

NET "LCD_CIKIS(10)" LOC = "M18" | IOSTANDARD = LVCMOS33 | DRIVE
= 4 | SLEW = SLOW ;
NET "LCD_CIKIS(9)" LOC = "L18" | IOSTANDARD = LVCMOS33 | DRIVE =
4 | SLEW = SLOW ;
NET "LCD_CIKIS(8)" LOC = "L17" | IOSTANDARD = LVCMOS33 | DRIVE =
4 | SLEW = SLOW ;
NET "LCD_CIKIS(0)" LOC = "R15" | IOSTANDARD = LVCMOS33 | DRIVE
= 4 | SLEW = SLOW ;
NET "LCD_CIKIS(1)" LOC = "R16" | IOSTANDARD = LVCMOS33 | DRIVE
= 4 | SLEW = SLOW ;
NET "LCD_CIKIS(2)" LOC = "P17" | IOSTANDARD = LVCMOS33 | DRIVE =
4 | SLEW = SLOW ;
NET "LCD_CIKIS(3)" LOC = "M15" | IOSTANDARD = LVCMOS33 | DRIVE
= 4 | SLEW = SLOW ;
NET "LCD_CIKIS(4)" LOC = "M16" | IOSTANDARD = LVCMOS33 | DRIVE
= 4 | SLEW = SLOW ;
NET "LCD_CIKIS(5)" LOC = "P6" | IOSTANDARD = LVCMOS33 | DRIVE =
4 | SLEW = SLOW ;
NET "LCD_CIKIS(6)" LOC = "R8" | IOSTANDARD = LVCMOS33 | DRIVE =
4 | SLEW = SLOW ;
NET "LCD_CIKIS(7)" LOC = "T8" | IOSTANDARD = LVCMOS33 | DRIVE =
4 | SLEW = SLOW ;
```

# 10. ÖZGEÇMİŞ


| | | |
|---|---|---|
| **Adı Soyadı** | **:** | Mert SONGEL |
| **Doğum Yeri ve Tarihi** | **:** | Malatya 1990 |
| **Lisans Üniversite** | **:** | Bolu Abant İzzet Baysal Üniversitesi |
| **Elektronik posta** | **:** | mertsongel@gmail.com |
| **İletişim Adresi** | **:** | İzzet Baysal mah. Gazi Mustafa Kemal Bulvarı Evim sitesi A blok No:36 Daire:10. |


**Yayın Listesi** **:**SONGEL M., AYTAR O., (2014), "Mikrodenetleyicili Sistem Aracılığıyla Ortamdaki Oksijen Konsantrasyonunu, Nemi ve Sıcaklığı Ölçmek", 3e ELECTROTECH Aylık Enerji, Elektrik, Elektronik Teknolojileri Dergisi, (sf. 168-174),SAYI:243.