YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

USB CİHAZ ARAYÜZÜNE AİT PROTOKOL KATMANI TASARIMI

Müh. Burcu KAPANOĞLU

F.B.E Elektronik ve Haberleşme Anabilim Dalı Elektronik Programında
Hazırlanan

YÜKSEK LİSANS TEZİ

Tez Danışmanı : Doç.Dr. Tülay YILDIRIM

Jüri Uyesi: Prof. Dr.Atilla ATAMAN

Jüri Uyesi: Prof.Dr.Galip CANSEVER

İSTANBUL, 2001

İÇİNDEKİLER Sayfa

KISALTMA LİSTESİ

ACK	Acknowledge Packet
ASIC	Application Specific Integrated Circuit
CRC	Cyclic Redundancy Checks
DVD	Digital Video Disk
EOP	End of Packet
FIFO	First In First Out
FPGA	Field Programmable Gate Array
GND	Ground
HDL	Hardware Description Language
HID	Human Interface Device
I/O	Input-Output
IEEE	Institute of Electrical and Electronics Engineers
IRP	I/O Request Packet
IRQ	Interrupt Request
ISA	Industry Standart
İTU	İstanbul Teknik Üniversitesi
LAN	Local Area Network
LSB	Least Significant Bit
MSB	Most Significant Bit
NAK	No Acknowledge Packet
NRZI	Non Return to Zero Invert
PBX	Private Branch Exchange
PC	Personal Computer
PCI	Peripheral Companents Interconnect
PID	Packet Identification
RAM	Random Access Memory
ROM	Read Only Memory
RTL	Register Transfer Level
USB	Universal Serial Port
VLSI	Very Large Scale Integration

ŞEKİL LİSTESİ

Sayfa

ÇİZELGE LİSTESİ

Sayfa

ÖNSÖZ

İTU ETA ASIC Tasarım Merkezinde proje gerçekleştirmemiz için bize imkan sağlayan değerli hocam Prof. Dr. Atilla Ataman'a, fikirleriyle ve tavsiyeleriyle her zaman yanımda olan sevgili hocam Doç. Dr. Tülay Yıldırım'a ve projenin başından sonuna kadar yardımlarını hiçbir zaman esirgemeyen, sorduğum soruları cevapsız bırakmayan proje yürütücümüz Arş. Gör. Sertaç Artan'a teşekkürlerimi sunarım.

Ayrıca manevi desteğini her zaman hissettiğim ailem ile Burçin Erkmen'e teşekkür ederim.

ÖZET

Günümüzde USB sistem mimarisini destekleyen çevresel cihazların kullanımı gittikçe yaygınlaşmıştır. Klasik uygulamaların getirdiği yetersiz sistem kaynakları, düşük performans ve konfigürasyon aşamasındaki zorluklar ve sorunlar, USB protokolünü destekleyen çevresel cihazların geliştirilmesi ile en aza indirilmiştir. Yüksek performans, düşük güç tüketimi, bir bağlantı arabirimine aynı anda birçok cihazın bağlantısının sağlanması ve otomatik konfigürasyon gibi beraberinde getirdiği avantajlar ile USB, yakın gelecekte tek çözüm olma yolunda ilerlemektedir.

Bu tezde, öncelikle USB sistem mimarisi içinde görev alan yazılım ve donanım elemanları tanıtılmış ve bu elemanların birbirleriyle olan fiziksel ve mantıksal ilişkileri açıklanmıştır. Tasarımı hedeflenen USB cihaz arayüz biriminin ana bloklarından biri olan ve kod düzeyinde tasarımı gerçekleştirilen protokol katmanının blok diyagramı verilmiş olup, her bir bloğun görevi ayrıntılarıyla açıklanmıştır.

Bu çalışmada, ITU ETA ASIC tasarım merkezinde yürütülmekte olan 'USB cihaz arayüzü' projesindeki, Verilog HDL tanımlama dili kullanılarak tasarlanan protokol katmanı açıklanmıştır. Tasarımda düşük hızlı (Low-speed) ve orta hızlı (Full-speed) çalışmalara ilişkin tanımları kapsayan USB 1.1 spesifikasyonu temel alınmıştır. Tasarlanan modüller Synopsys-Design-Analyzer sentezleyici programı kullanılarak sentezlenmiştir. Kod düzeyinde tasarlanan blokların davranışlarının, test edilebilmesi için uygun test modülleri hazırlanmıştır. Cadence DFII (Design Framework II) temel lojik simülatörü Verilog XL kullanılarak tasarımın doğruluğu hazırlanan test modülleri ile sınanmıştır. Protokol katmanının veri alma ve veri gönderme işlemleri sırasındaki veri akışını gösteren simülasyon sonuçları ekte sunulmuştur. Bir sonraki aşamada, tasarım FPGA ortamına aktarılmış ve test kartı üzerinde Logic Tester (LV500) ve bir PC yardımıyla sistem gerçek ortamda test edilmektedir.

Anahtar Kelimeler: USB, Protokol katmanı, USB sistem mimarisi, USB cihaz arayüzü

ABSTRACT

Today the peripheral devices supporting USB system architecture are frequently in use. By the development of peripheral devices supporting USB protocols, it becomes possible to overcome the difficulties which originate from the unsatisfactory system resources, low performance and shortcomings in configuration process appearing due to classical applications. With many advantages like high performance, minimum power consumption, opportunity to ensure simultaneous connections for the multiple devices and automatic configuration, USB seems to be a unique solution in the near future.

In this work, first of all, the software and hardware components taking part in the USB system architecture are introduced and the physical and logical relationships of these components with each other are explained. The design of the protocol layer, which is one of the main blocks in USB device interface, is realized in HDL code-level and the block diagram of it is given with the description of each block in detail.

In this thesis; information about the concerning blocks which are designed by using Verilog HDL will be submitted. This project of USB interface is being realize at ITU ETA ASIC design center. This design is based on the USB 1.1 specification which describes the low speed and full speed operations. The code-level designed modules have been synthesized with the help of the synthesizer program – Sysnopsys Design Analyzer. Appropriate test modules are designed to test the behavior of the designed modules. The accuracy of the design is tested by using the basic logic simulator Verilog XL of Cadence DF II (Design Framework II). In the last section of this thesis, simulation results belong to data transfer between protocol layer and physical layer, and between the protocol layer and USB device, during both receive and transmit operations, will be presented. The next step is, transferring the design to an FPGA board and testing the overall system in a real enviroment with the help of a Logic Tester and a PC.

Keywords: USB, Protocol layer, USB system architecture, USB device interface

1. GİRİŞ

USB (Universal Serial Bus) protokolü, kişisel bilgisayarlar veya iş istasyonları ile çeşitli çevre birimleri arasında haberleşmeyi sağlayan bir seri haberleşme protokolüdür. USB sistem mimarisini oluşturmaktaki asıl amaç; çevresel cihazların bilgisayara bağlantısını kullanıcı açısından kolaylaştırmak, bilgisayara aynı anda mümkün olduğunca çok sayıda cihaz bağlantısı sağlayabilmek ve veri iletimindeki performansı arttırmaktır. Klasik uygulamaların getirdiği yetersiz sistem kaynakları, düşük performans ve konfigürasyon aşamasında meydana gelen sürücü ile ilgili problemler USB mimarisi ile elimine edilmiştir.

Günümüzde USB kullanımı son derece yaygınlaşmış ve pek çok ürün dalında yüzlerce şirket USB çevresel birimler üretmektedir. Bu ürünler arasında fare (mouse), klavye, oyun araçları (Joystick), sayısal telefon, kamera, modem, CD-ROM sürücü, floppy sürücüler, ses kartları, ses yongalı hoparlörler, ISDN modemler, ağ ürünleri sayılabilir.

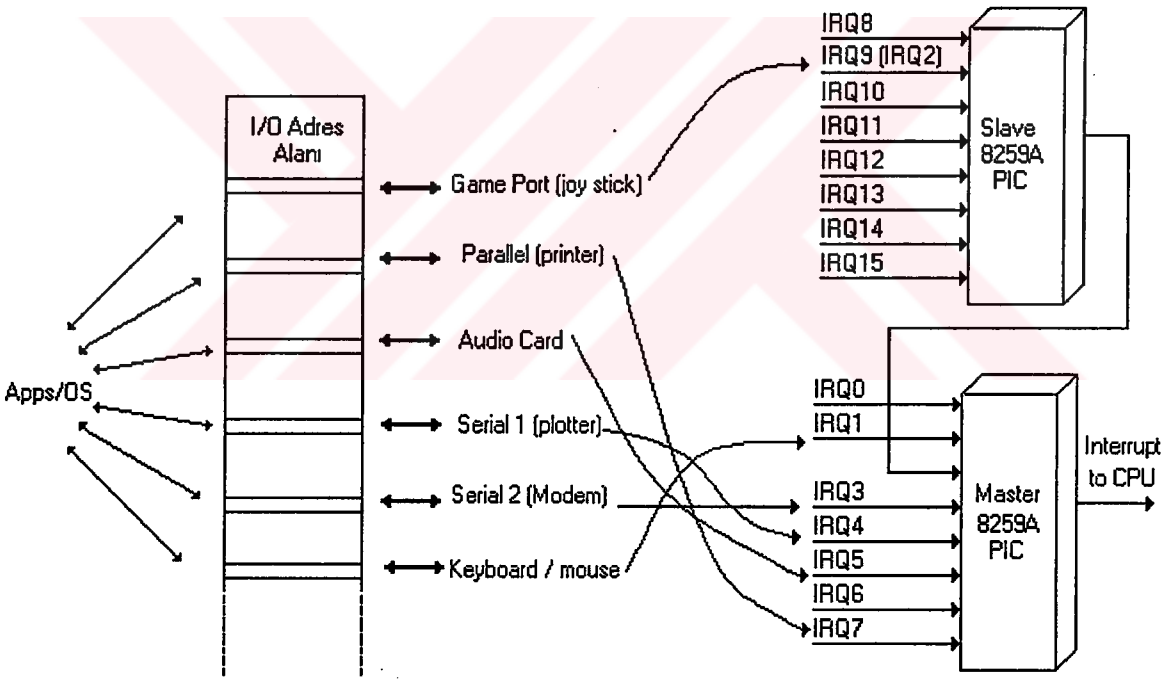
Bu tezde öncelikle USB sistem mimarisi tanıtılacak, USB haberleşme sistemini oluşturan elemanlar ve bu elemanların birbirleri ile olan ilişkileri açıklanacaktır. İTU ETA ASIC tasarım merkezinde yürütülmekte olan USB cihaz arayüzü gerçekleştirme projesinde tasarlanan bloklar hakkında bilgi verilecek ve bu tasarıma ilişkin ekte simülasyon sonuçları sunulacaktır.

1.1 Tarihsel Gelişim

Bu bölümde, çevresel cihaz ile kişisel bilgisayarın haberleşmesini sağlayan, farklı protokol ve performansları destekleyen veri yolu (Bus) mimarilerinin tarihsel gelişimi incelenecektir. Günümüzde popülerliği gittikçe artan USB mimarisinin, modern PC oluşumunda ne derece önemli olduğu anlatılacaktır.

IBM ilk bilgisayarı ürettiğinde PC (kişisel bilgisayarlar), çevresel birimlerle haberleşme sorunları yaşıyordu. Üreticiler arasında, çevresel donanım üretilmesi için herhangi bir standart yoktu. Örneğin iki farklı üreticinin kartları aynı PC içinde çalışmayabiliyordu. 1990'ların başında ISA (Industry Standart Architecture) slot için standartlar belirlendi. Ne yazık ki bilgisayara birden fazla ISA mimarisi destekleyen çevresel birim bağlandığında, IRQ kesme hatları (interrupt line) ve diğer sistem kaynakları arasında çakışma ve uyumsuzluk sorunu yaşanıyordu. Aynı zamanda performansı gittikçe artan cihazlar için band genişliği de yetersiz kalıyordu. Daha iyi çözüm arayışları başlamıştı. Birçok PC ve PC bileşen satıcısı daha yüksek band genişliğe sahip sistem için birlikte çalıştılar. ISA'daki sistem kaynaklarının çakışma

sorununu da azaltan sistem PCI (Peripheral Components Interconnect) olarak adlandırıldı. Otomatik konfigürasyon için yazılım desteği Windows işletim sistemine eklendi. Böylece Tak ve Çalıştır kavramının (Plug and Play) başlangıcı olan PCI kartların kolay kurulumu sağlandı. Günümüzde yaygın olarak kullanılan PCI kartları ilk başlarda başarılı olarak görülse de basit I/O cihazları için gereksiz komplekslilik ve yüksek maliyet teşkil etmektedir. Başka bir dezavantajı da ISA'da olduğu gibi kasa içinde olmasıdır. Bu da cihazın kurulumunu zorlaştıran bir etkendir. Şekil 1.1'de klasik mimariler ile uyumlu çalışan çevresel cihazların PC'ye eklenmesi ile I/O adres alanının ayrılması ve atanılan IRQ hatları görünmektedir. PC'ye eklenen çevresel cihaz sayısının artması ile çakışma ve uyuşmazlık sorunun kullanıcının karşısına çıkması aşikardır. (Hyde,1999)



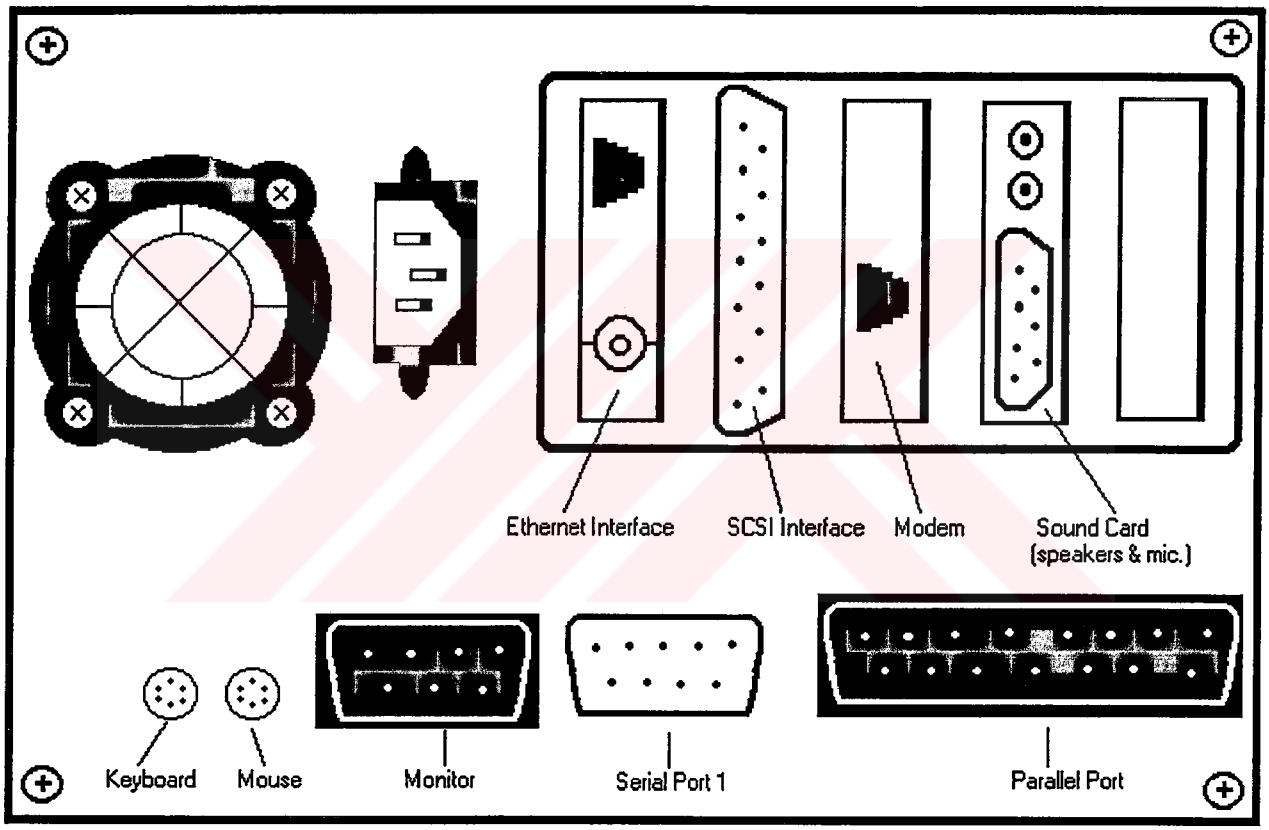
Şekil 1.1 Klasik mimarilerde adres alanı ve IRQ hatlarının kullanımı (Anderson,1999)

Klasik mimariler (PCI, EISA, ISA) ile uyumlu standart çevresel birimlerin bilgisayar ile ara bağlantı noktaları, aynı anda sadece bir cihazın bağlantısını destekler. Yeterli bağlantı birimine sahip olmayan bilgisayara cihaz bağlamak için genişletme kartına ihtiyaç duyulur. Genişletme kartı kullanarak bu sorun ortadan kaldırılabilir. Fakat pahalı ve performansı düşüren bir çözümdür.

Fare, klavye, yazıcı, harici modem, monitör vb. standart çevresel cihazların bağlantı noktaları birbirinden oldukça farklıdır. Farklı bağlantı tipleri ve kablolar karmaşıklığı ve kullanımdaki

zorluğu arttıran diğer bir faktördür. PC'nin arka düzleminde bulunan, farklı cihazlar için kullanılan ara bağlantı birimleri Şekil 1.2'de gösterilmiştir.

Klasik mimariyi destekleyen herhangi bir çevresel cihaz PC'ye ilk bağlandığında disketten bu cihaza ait konfigürasyon bilgilerinin okutulması gerekir. Bilgisayar hakkında fazla bilgi sahibi olmayan kullanıcıya konfigürasyon işlemleri karmaşık gelebilir. Donanımsal ve yazılımsal kurma işlemlerinin sonunda sistem kaynakları arasında uyuşmazlık yaratılamadığı takdirde cihaz kullanıma hazırdır.



Şekil 1.2 PC'nin arka panelinde bulunan, farklı cihazlar için kullanılan ara bağlantı birimleri

Konfigürasyon aşamasındaki zorluklar, sistem kaynaklarındaki çatışma veya uyuşmazlıklar, farklı bağlantı arabirimlerinin ve kablo tiplerinin getirdiği maliyet farkı ve bağlantılardaki karmaşıklıklardan dolayı üretici firmalar tamamen kullanıcı isteklerini karşılayacak yeni bir sistem mimarisi tasarlanması için bir araya geldi. Amaç klasik mimarilerin getirdiği sorunları ve karmaşıklığı ortadan kaldıracak çözüm yollarına gitmekti.

Yeni sistem mimarisi ile ;

- herhangi bir çevresel cihazın bilgisayara tek tip bağlantı arabirimi (connector) ile bağlantısını sağlamak mümkün olmalıdır.

- tek bağlantı noktası vasıtasıyla aynı anda birçok cihazın bilgisayarla bağlantısı sağlamak mümkün olmalıdır.
- konfigürasyon işlemleri diskete ihtiyaç olmaksızın otomatik olarak yapılabilmelidir.
- güç tüketimi oldukça düşük olmalıdır.
- sistemin performansı günümüz çevresel cihazlarına cevap verebilmelidir.

Çizelge 1.1'de günümüz çevresel cihazların uygulamada gereken performansları verilmiştir.

Çizelge1.1 Günümüz çevresel cihazların uygulamada gereken performansları

Performans	Uygulamadaki Cihazlar
Düşük Hızlı (Interactive Cihazlar)	Klavye Fare Oyun aygıtları Monitor
Orta Hızlı (Telefon, Audio)	ISDN PBX Digital Ses uygulamaları Tarayıcı,Yazıcı
Yüksek Hızlı (Video , LAN)	Video konferans uygulamaları PnP LAN Digital Kamera

Çevresel cihazların performans gereksinimleri dikkate alınarak daha önce bahsedilen klasik mimarilerin getirdiği dezavantajları ortadan kaldırmak amacıyla farklı karmaşıklık ve performansta çeşitli mimariler geliştirilmiştir. Bu mimarilere ilişkin bus isimleri ve performanslar Çizelge 1.2'de verilmiştir.

Acces bus mimarisi 100Kb/s hız ile uygulamada geniş band genişliği gerektiren çevresel cihazlar için uygun bir çözüm değildir.

Geoport mimarisi, sadece telekomünikasyon uygulamalarında kullanıldığından PC çevresel cihazları ile uyumlu değildir.

400Mb/s hıza sahip Fireware mimarisini destekleyen cihazlar daha çok yüksek kalite video, ağ, sabit disk ve DVD gibi yüksek band genişliği isteyen sistemler için uygun görülmüş fakat

karmaşıklığı ve maliyeti yönünden düşük performanslı cihazlar tarafından tercih edilmez.

USB, 1995 yılında Compaq, Hewlett Packard, Intel, Lucent, Microsoft, NEC ve Philiphs firmalarının önderliğinde geliştirilmiş sistem mimarisidir. USB 2.0 2000 yılında standartlaştırılmasıyla birlikte hız, performans ve maliyet yönünden diğer mimarileri geride bıraktı. USB 1.1'den 40 kat hızlı olan USB 2.0 mimarisi, USB 1.1 ile uyumlu üretilmiştir. USB 2.0 sistemleri, USB 1.1 ile uyumlu cihazları çalıştırabilecek fakat tersi mümkün olmayacaktır.

Çizelge 1.2 Geliştirilen farklı mimariler ve getirdikleri performanslar (Anderson,1999)

Bus Adı	Hız (Data Rate)
Access Bus	100Kb/s
GeoPort	2.048Mb/s
IEEE 1394 (Firewire)	400Mb/s
USB 1.1 USB 2.0	12Mb/s 480Mb/s

USB arabirimine ilk destek Microsoft tarafından kısmen de olsa Windows 95 'in ilk sürümü OEM2.1 ile verildi. DOS ve Windows NT de USB desteği yoktur. Windows 98, 2000 ve ME, MacOS işletim sistemlerinde ise USB tam desteği verilmektedir.

Bundan sonraki bölümlerde USB1.1 protokolünde (specification) yer alan tanımlar dahilinde USB sistem mimarisi incelenecektir.

1.2 USB Sistem Mimarisinin Özellikleri

USB sistem mimarisinin özellikleri, beraberinde getirdiği avantajlar ve diğer klasik mimarilere göre bazı üstünlükleri aşağıda sıralanmıştır. Güç, hız, maliyet, kullanım kolaylığı gibi kriterler esas alınarak bu özellikler çıkarılmıştır.

1.2.1 Tak ve Çalıştır (Plug and Play)

USB cihaz PC'ye bağlandığı zaman otomatik olarak sistem tarafından bulunur. Kullanıcının dışarıdan müdahale etmesine gerek kalmadan USB sistemi, cihazı konfigüre eder.

1.2.2 Tek Tip Bağlantı Arabirimi

Tüm USB cihazların PC'ye bağlantı noktaları mekanik ve elektriksel olarak aynı yapıdadır. Tek bir bağlantı arabirimine birden fazla cihazın bağlantısı USB Hub ile sağlanır.

1.2.3 127 Cihaz Bağlantısı

Her bir USB sistemi, USB Hub vasıtası ile aynı anda 127 USB cihaz bağlantısı destekleyebilir.

1.2.4 Düşük Maliyet

USB sistemi PC'ye bağlanan çevresel cihazlar için tek tip bağlantı birimi ve performansı dikkate alındığında düşük maliyet sağlayan bir çözümdür.

1.2.5 Düşük, Orta ve Yüksek Hızlı Cihazlar

USB sistemi düşük hızlı 1.5Mb/s (Low speed), orta hızlı 12Mb/s (Full speed), yüksek hızlı 480Mb/s (USB 2.0 : High speed) cihazları destekler.

1.2.6 USB Kablodan Güç Eldesi

Çevresel USB cihazlar direk olarak USB kablodan beslenebilirler. 5V DC gerilimi kablodan sağlayabilirler. Akım ise USB hub portlarına bağlı olarak 100mA ile 500mA arasında değişebilir.

1.2.7 Sistem Kaynakları

Sisteme bağlı olan USB cihazlar, sistem kaynaklarını direk olarak kullanmazlar. Sistem, klasik yapılardan (PCI, ISA, EISA) farklı olarak hatta bağlı USB cihaz sayısı ne olursa olsun tek bir IRQ hattını ve I/O adres alanını kullanır.

1.2.8 Veri İletiminde Hata Tespiti ve Hata Düzeltme Mekanizması

USB haberleşme protokolü hatasız veri iletimini sağlamak üzere, veri üzerinde birden fazla kontrol işlemini destekler. Hata meydana geldiğinde aynı veri tekrar iletilir.

1.2.9 Güç Koruma Özelliği

USB cihazları hatta 3ms süresince veri iletimi olmadığı belirlendiğinde bekleme durumuna (suspend) geçerler. Bu durumda hattan 500μA'den fazla akım çekilmez. Böylece gereksiz yere güç tüketilmez.

1.2.10 Sisteme Bağlanma ve Sistemden Ayrılma

USB cihazlar herhangi bir zamanda sisteme bağlanıp, sistemden ayrılabilme (Hot Pluggable) özelliğine sahiptirler. Örneğin, bilgisayar 'on' konumundayken herhangi bir USB cihaz bağlandığında sistem tarafından tanınır.

1.2.11 Dört Farklı Transfer Tipi

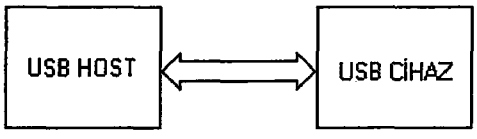
USB cihazların ve sistemin ihtiyaçlarına bağlı olarak kontrol, kesme (Interrupt), yığın (Bulk), isochronous olmak üzere 4 farklı transfer tipi tanımlar. Her transfer tipi farklı karakteristiklere sahiptir.

2. USB SİSTEM MİMARİSİ

Bu bölümde USB sistem mimarisini oluşturan donanımsal (hardware) ve yazılımsal (software) birimler hakkında genel bilgi verilecektir. Ayrıca bu birimlerin sistem içinde birbirleriyle olan etkileşimleri, mantıksal veya fiziksel bağlantıları ve sistem içindeki görevleri incelenecektir.

2.1 USB Yazılım ve Donanım Birimleri

Ana bilgisayar (Host) ile çevresel cihaz arasındaki haberleşme akışını sağlayan USB sistemi en genel ifadeyle Şekil 2.1'de gösterilmiştir.



Şekil 2.1 En genel halde USB sistemi (Compaq,1998)

USB Host sisteme bağlı tüm USB cihazlarını yönetmekle sorumludur. Host ile cihaz arasındaki haberleşme akışını başlatan tüm veri paketleri, host tarafında bulunan yazılım ve donanım birimleri vasıtasıyla oluşturulur. USB sistemini oluşturan başlıca yazılım ve donanım elemanları aşağıda belirtilmiştir.

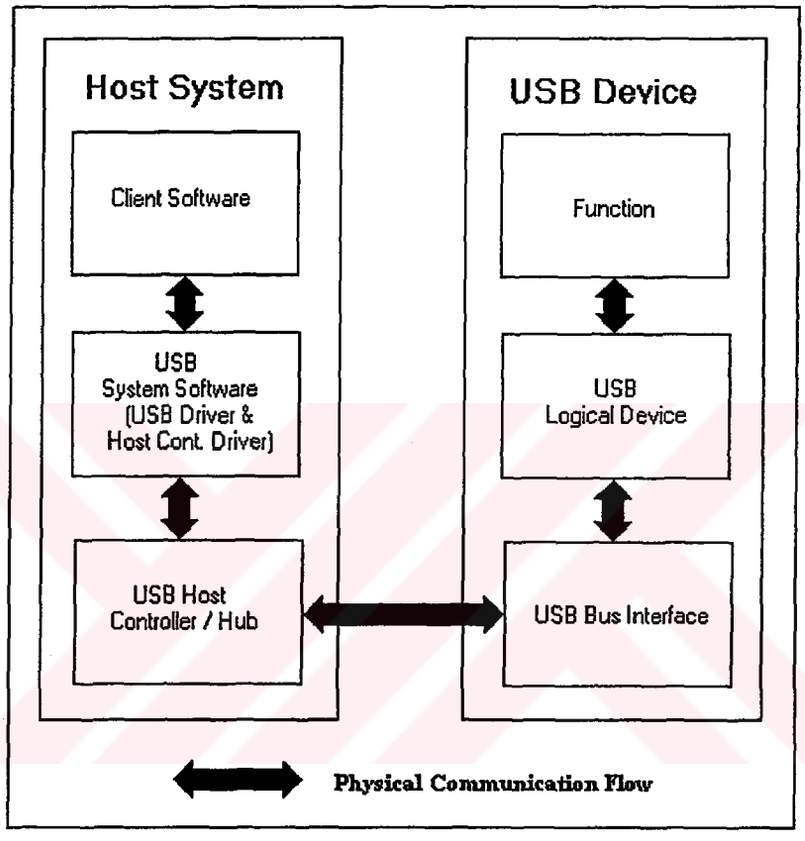
- USB Donanım Elemanları
  - USB Host Denetleyici / Kök (Root) Hub
  - USB Hub
  - USB Cihazlar

- USB Yazılım Elemanları
  - USB Cihaz Sürücüler (Client Software)
  - USB Sürücüsü
  - Host Denetleyici Sürücüsü

Takip eden bölümde sistemi oluşturan bu elemanların USB haberleşmesindeki rolleri anlatılacaktır. Bu bölümleri okurken Şekil 2.2'yi dikkate alınız. Host ile cihaz arasında arabirim oluşturan USB bus arayüzü, elektriksel işaretlerin iletimini gerçekleştiren fiziksel bir elemandır. Bölümün sonunda fiziksel arayüzü oluşturan USB kablo ve USB konektör yapısından ayrıntılı bahsedilecektir.

2.1.1 USB Cihaz Sürücüler

USB Cihaz Sürücüleri, USB sürücüsüne isteklerini I/O istek paketleri (IRP) vasıtasıyla gönderen yazılım birimidir. IRP'ler cihazdan hosta ve ya hosttan cihaza veri iletimini başlatan özel istek paketleridir. USB cihaz sürücüleri, bağlı olduğu cihaz ve cihaza ait fonksiyonların özelliklerine (band genişliği ve bus erişim periyodu) bağlı kalarak istek paketlerini hazırlar.



Şekil 2.2 USB sistemi içindeki haberleşme akışı (Compaq,1998)

2.1.2 USB Sürücüsü

USB sürücüsü, USB cihaz sürücüsü ile USB host denetleyicisi arasında arabirim sağlar. Bu yazılım, USB cihaz sürücüsünün isteklerini USB cihaza veya USB cihazdan yönlendirilen bir veya daha fazla sayıda işlem birimine (transactiona) çevirir. İşlem birimleri veri paketlerinin bir araya gelmesi ile oluşur ve belirli bir amaca hizmet ederler. USB sürücüsü, konfigürasyon sırasında USB cihaz tanımlayıcılarından aldığı bilgiler doğrultusunda hedef USB cihazının karakteristiklerini ve bu cihazla USB vasıtasıyla nasıl haberleşeceğini bilir. USB cihazın ihtiyaçlarını, sınırlamalarını (zaman, paket boyutu. vb..) dikkate alarak cihaz ile iletişimi sağlayan işlem birimlerini belirler.

2.1.3 USB Host Denetleyici Sürücüsü

USB host denetleyici sürücüsü, USB cihaza gönderilmesi gereken işlem birim listesini oluşturmakla görevlidir. Bu liste USB hatta bağlı bir veya daha fazla cihaza yönlendirilmiş askıda bekleyen (pending) işlem birimlerinden oluşur. İşlem birimi veya çerçeve (frame) listesi, her 1ms'de işlenecek işlem birimi sırasını iletimin türüne ve ihtiyaçlarına bağlı olarak belirler. Çerçeve (frame) kavramı bu protokolde 1ms'lik süreyi ifade eder. USB host denetleyici bu işlem birimi listesini 1ms aralıklarla aktif eder. USB host denetleyici bekleyen işlem birimlerini kök hub vasıtasıyla USB hatta çıkarır. Her çerçeve, orta hızlı cihazlar için anlamı olan ve çerçevenin başını gösteren (SOF:start of frame) bir işlem birimi ile başlar ve o listedeki diğer işlem birimlerinin yayınlanmasıyla devam eder.

2.1.4 USB Host Denetleyici & Kök Hub

USB üzerindeki tüm haberleşme host'un kontrolü altındadır. Host donanımı, USB sistem üzerinde işlem birimleri başlatan USB host denetleyici ve USB cihazlar için bağlantı noktaları sağlayan kök hub'dan oluşur.

2.1.4.1 Host Denetleyici

Host denetleyici, host yazılımı tarafından hazırlanan işlem birimlerini transfer tanımlayıcısında belirtilen veri yapısına bağlı olarak yaratmakla görevlidir. Transfer tanımlayıcısında bulunan bilgiler, hedef USB cihazın adresini, transferin tipini ve yönünü içerir.

Cihaza bilgi yazma işlemi için gereken veri, hafıza yazmaçlarından okunur. Host denetleyici tarafından, veri üzerinde paralelden seriye dönüştürme işlemi gerçekleştirildikten sonra seri veri paketi hatta çıkarılmak üzere kök hub'a gönderilir. Cihazdan bilgi okuma işlemi içinde benzer işlem yapılır. Bu sefer host denetleyici, okuma işlem birimi oluşturur ve hatta çıkarmak üzere kök hub'a gönderir. Veri yapısında bulunan adres bilgisinden hedef cihaz belirlenir. Hedef cihaz gerçek veriyi önce kök hub'a çıkarır. Ardından host denetleyici tarafından veri üzerinde seriden paralele dönüştürme işlemi yapılarak, cihaz sürücünün hafıza hücrelerine iletilir.

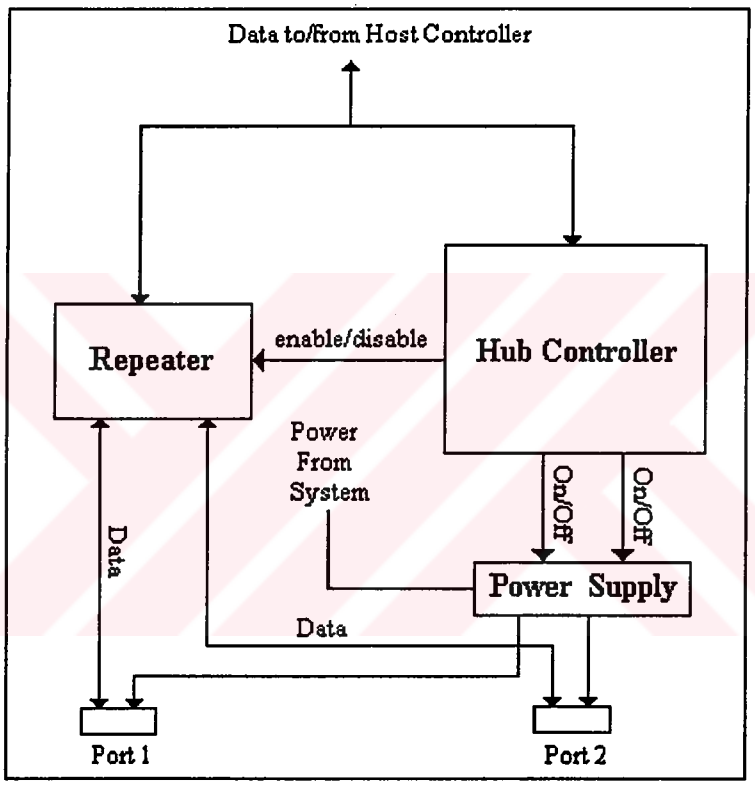
2.1.4.2 Kök (Root) Hub

Host denetleyici tarafından üretilen işlem birimleri USB hatta çıkarılmak üzere kök hub'a gönderilir. Sonuçta her USB işlem birimi kök hub'da başlar. USB cihazlar için ek bağlantı

noktası oluşturması özelliğinin yanında kök hub'ın diğer görevleri aşağıda sıralanmıştır.

- USB portlarına verilen gücü kontrol eder
- Portları aktif (enable) ve pasif (disable) edebilme özelliğine sahiptir.
- Herhangi bir portuna takılan cihazları algılayabilme özelliğine sahiptir.

Kök Hub, Şekil 2.3'te görüldüğü gibi host denetleyici ve tekrarlayıcı (Repeater) birimlerinden oluşur. Hub denetleyici, hub'a yapılan erişimlere cevap verir. Tekrarlayıcı ise, yönlendirilmiş işlem birimlerini iletme görevi üstlenir.



Şekil 2.3 Kök hub fonksiyonunun en genel haliyle blok diyagramı (Anderson,1999)

2.1.5 USB Hub

Kök hub'a ek olarak, bir çok cihazın bağlanmasıyla USB sistemin genişlemesi ilave hub'lar ile sağlanır. USB hub'lar tek başına cihaz olarak sistem içinde bulunabildiği gibi, klavye ve monitör gibi cihazlara entegre edilmiş olarak da sistem içinde yer alabilir. Bu cihazlara "birleşik cihaz" (compound device) adı verilir. (Şekil 2,4)

USB hub'lar, kök hub'da da bahsedildiği gibi iki ana fonksiyonel elemandan oluşur.(Şekil2.5)

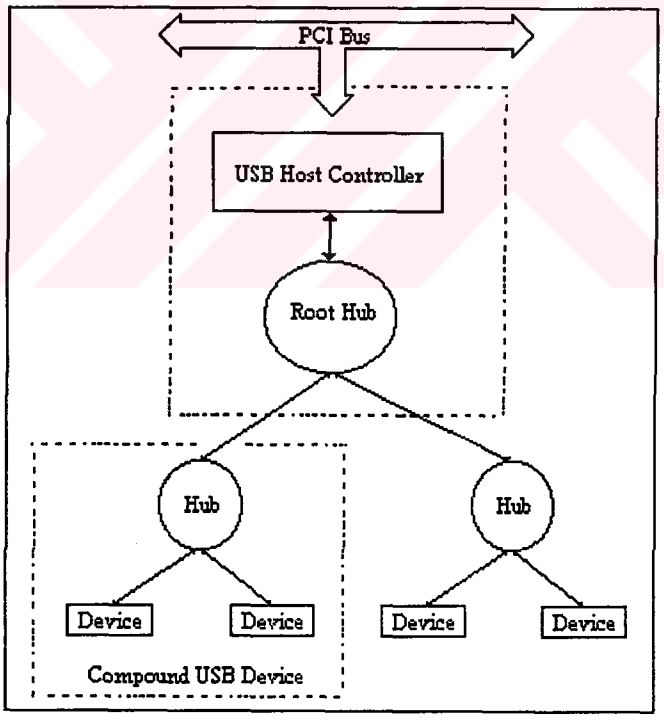
- Hub Denetleyici
- Tekrarlayıcı (Repeater)

2.1.5.1 Hub Denetleyici

Host'un hatta bağlı olan cihazı hub olarak tanıyabilmesi için gerekli olan tanımlayıcıları içerir. Hub denetleyici, hub durum bilgilerini; cihaz bağlama, bağlantı koparma ve diğer durum bilgilerinin tespit edilebilmesi için hafızasında tutar. Denetleyicinin bir diğer görevi de hub işlemlerinin kontrolü için host yazılımından komut almaktır. Örneğin portların aktif edilmesi, gücün sağlanması vb.

2.1.5.2 Hub Tekrarlayıcı (Repeater)

Hub'a ulaşan veri trafiği yukarı akış (upstream; host'a doğru veri akışı) veya aşağıya akış (downstream; alt birimlere doğru veri akışı) olmak üzere iki yönde iletilmelidir. Host tarafında başlatılan veri transferi hub'ın kök portuna varacak ve aşağıya akış yönünde bulunan tüm aktif portlara iletilecektir. Hedef cihaz ise, host tarafından başlatılan veri transferini yukarı akış yönünde cevaplayacaktır.(Şekil 2.6)

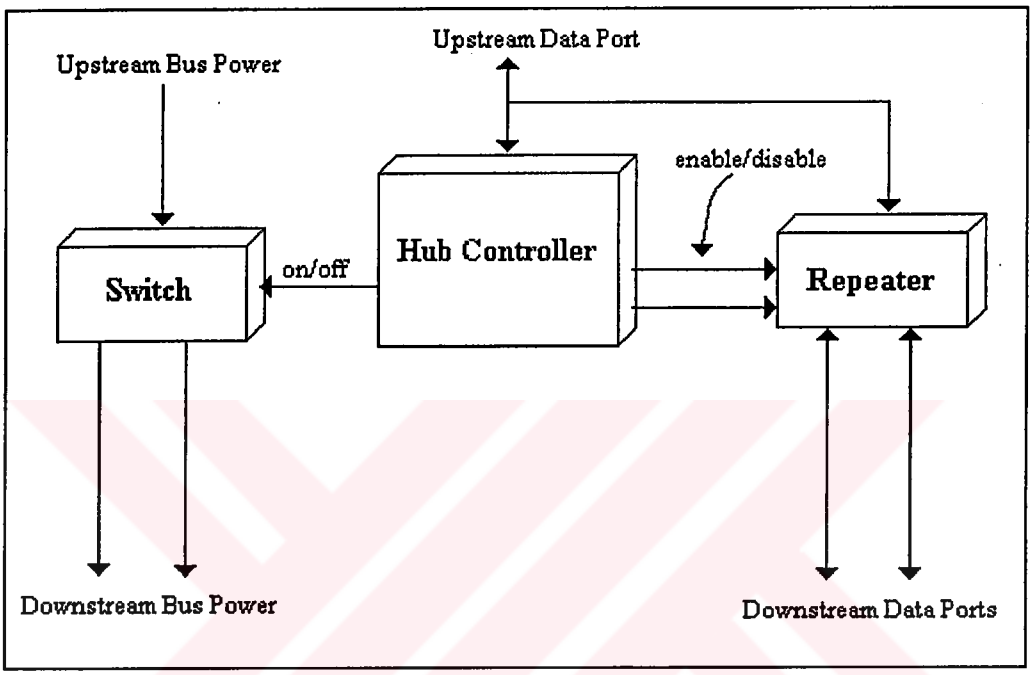


Şekil 2.4 Sistem içinde yer alan farklı hub tipleri (Anderson,1999)

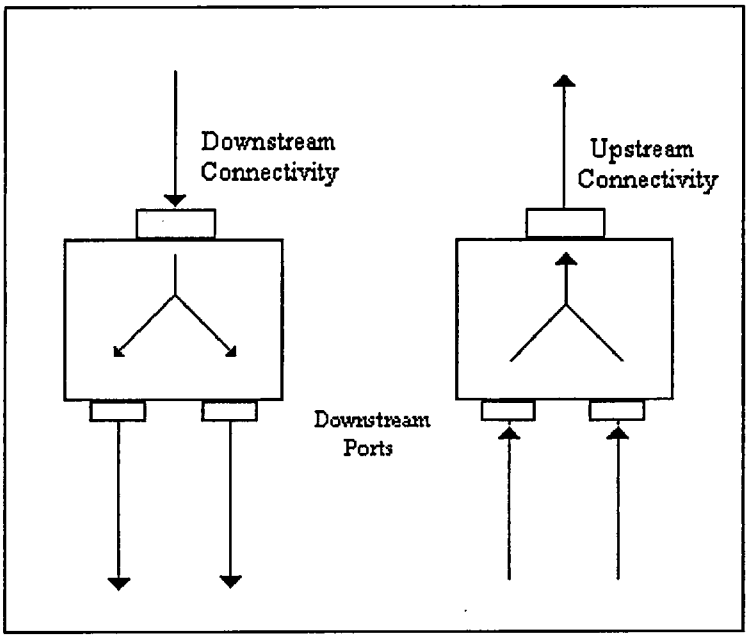
2.1.6 USB Cihazlar

USB cihazlar kendi özelliklerini, karakteristiklerini ve yeteneklerini host'a betimleyen cihaz tanımlayıcı bilgilerine sahiptir. Bu bilgilere cihazın konfigürasyonu sırasında host tarafından

ihtiyaç duyulur. Ayrıca USB cihaz sürücüsü, tanımlayıcılar vasıtası ile bağlı olduğu cihaza uygun bir şekilde erişim sağlayabilir. USB 1.1 protokolünü destekleyen USB cihazları hızlarına bağlı olarak iki guruba ayrılır. Hızlarının dışında bazı karakteristikleri de birbirlerinden oldukça farklıdır. Takip eden bölümde orta ve düşük hızlı cihazlara ilişkin daha ayrıntılı inceleme yapılacaktır.



Şekil 2.5 USB Hub'ı oluşturan başlıca elemanlar (Anderson,1999)



Şekil 2.6 Hub tekrarlayıcı biriminin aşağı-yukarı akış yönündeki davranışı (Anderson,1999)

2.1.6.1 Orta Hızlı Cihazlar

Orta Hızlı Cihazlar, USB hattı üzerinde yayılan ve kendi adreslerine ulaşan tüm işlem birimlerini algılayıp işlerler. Bu cihazlar seri veri alışverişini 12Mb/s hızda gerçekleştirir.

2.1.6.2 Düşük Hızlı Cihazlar

Düşük hızlı cihazlar sadece giriş paketinin (preamble packet) ardından gelen paketleri algılayıp işlerler. Düşük hızlı hub portları, orta hızlı veri iletimi süresince aktif edilmez (disable). Bu mekanizma ile orta hızlı bus trafiğinin düşük hızlı kablolar üzerinden gönderilmesi engellenmiş olur. Giriş paketi (preamble packet), ardından gelen verinin düşük hızda yayılacağını belirtir. Hub'lar hattan giriş paketini algıladıklarında düşük hızlı portlarını aktif eder

2.2 USB Haberleşme Akışı

USB cihaz sürücüsü, USB sistem yazılımını transfer talebi için çağırdığı zaman haberleşme akışı başlamış olur. Cihazda yer alan uç nokta ile USB cihaz sürücüsü arasında gerçekleşen her iletim haberleşme kanalları yolu ile taşınır. USB sistem yazılımı, USB cihaz sürücünün iletim talebini (IRP), USB protokol mekanizmasına ve ilgili cihazın band genişliği sınırlamalarına dayanarak işlem birimleri halinde düzenler. Bu işlem birimleri, cihazda bulunan tanımlayıcılara ve cihazın gereksinimlerine bağlı olarak USB hatta çıkarmak üzere USB host denetleyici sürücüsü tarafından listelenir. USB host denetleyici sürücüsü tarafından oluşturulan listeye dayanarak USB host denetleyicisi gerekli işlem birimlerini üretir. Talep edilen iletim yönüne dayanarak veri iletimi host tarafındaki yazmaçlardan USB cihaza ya da USB cihazdan host tarafındaki yazmaçlara gerçekleşir. USB sistem yazılımı, USB cihaz sürücüsüne iletimin tamamlandığına dair bilgi verir. USB cihaz tarafında bulunan her fonksiyon host'a, yazmaçların veya uç noktaların (endpoint) toplamı şeklinde görünür. Her uç nokta diğerlerinden farklı iletim karakteristiklerine sahiptir.

USB sistemi tarafından desteklenen iletim tipleri aşağıda sıralanmıştır.

- Isochronous Transfer (Gerçek Zamanlı)
- Yığın (Bulk) Transfer
- Kesme (Interrupt) Transfer
- Kontrol Transfer

İletim tiplerinin karakteristikleri hakkında genel bilgi 4. Bölümde sunulacaktır.

2.2.1 Band Genişliği ve Çerçeve (Frame) Bağıntıları

USB Cihaz sürücüleri, USB hat üzerinden belirli bir uç noktaya (endpoint) veri iletimi gerçekleştirmek istediği zaman iletimi başlatması için isteklerini USB sürücüye bildirirler. Aynı anda birçok cihaz aynı bus üzerinden haberleşme sağladığından, fonksiyonların karakteristiklerine dayanarak USB bus, cihazlar arasında dengeli olarak paylaştırılmıştır. İletim tipine bağlı olarak veri uzunluğu, belirli bir değer ile sınırlandırılmıştır. Bu nedenle herhangi bir USB cihaz yüksek band genişliğine sahip veri iletimini bir seferde gerçekleştiremez. Yüksek band genişliğine sahip veri paketleri, maksimum veri uzunluğu sınırlamasına bağlı olarak daha düşük band genişliği olan veri paketlerine bölünür. İletim USB hat üzerinden daha uzun periyotta fakat hattı meşgul etmeden sağlanır.

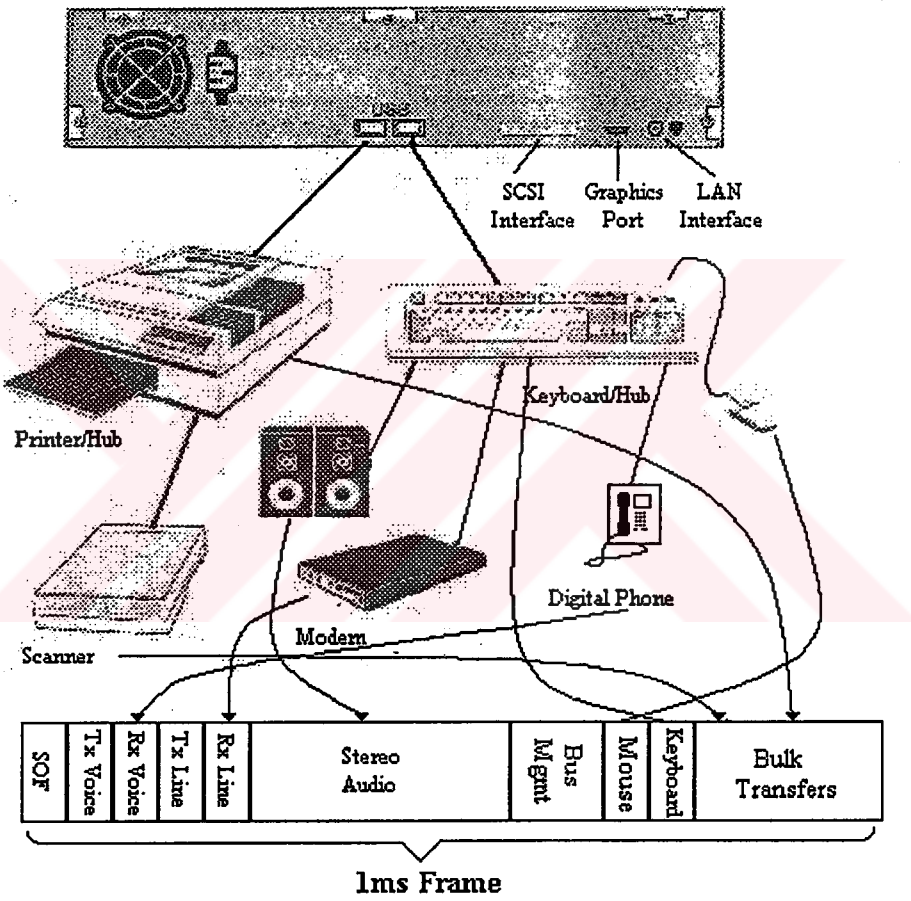
Veri iletimi her 1ms süresi içinde düzenli olarak gerçekleşir. Bu süre USB sisteminde 'çerçeve' (=frame) olarak adlandırılır. Her USB cihaz için gerekli band genişliği 1ms'lik çerçeve içinde dengeli olarak bölünmüştür. Band genişliğinin USB hatta bağlı olan çeşitli cihazlar arasında paylaştırılması, konfigürasyon aşamasında tanımlayıcılar yolu ile belirlenen cihazın karakteristiklerine bağlı olarak host tarafından yapılır. Cihaz tanımlayıcılar, gerekli band genişliğini konfigürasyon sırasında sistem yazılıma bildirirler. Sistem yazılımı diğer cihazlar arasında kullanılan band genişliğini kontrol eder . Eğer kullanılmayan band genişliği cihazın 1ms'lik çerçeve içinde talep ettiği maksimum band genişliğini karşılıyorsa cihaz konfigüre edilir. Talep edilen yeterli band genişliği yoksa eklenen cihaz konfigüre edilemeyecek ve bu durum kullanıcıya bildirilecektir.

Şekil 2.7'de USB sisteme bağlanan cihazlar topluluğu ve cihazların özelliklerine ve band genişliği taleplerine bağlı olarak bir çerçevenin paylaşımı görülmektedir. Gerçekte tüm cihazlar aynı çerçevede veri iletimi gerçekleştirmek zorunda değildir. Örneğin USB sistemine bağlı olan klavyede her n. çerçevede periyodik olarak kesme olup olmadığı host tarafından kontrol edilir.

2.2.2 Veri Paketleri ile İşlem Birimleri Arasındaki İlişki

Daha öncede belirtildiği gibi USB sürücüsü, host denetleyici sürücüsüne veri paketi gönderme talebini ilettiği zaman, host denetleyici sürücüsü ilgili cihazın karakteristiğine ve sınırlamalarına dayanarak veri paketini maksimum band genişliklerine bölerek çerçeveler arasında paylaştırır. Host denetleyici konfigürasyon sırasında okuduğu tanımlayıcıları yorumlar ve 1ms içinde gönderilecek veri paketleri listesi host denetleyici sürücüsü tarafından hazırlanır.

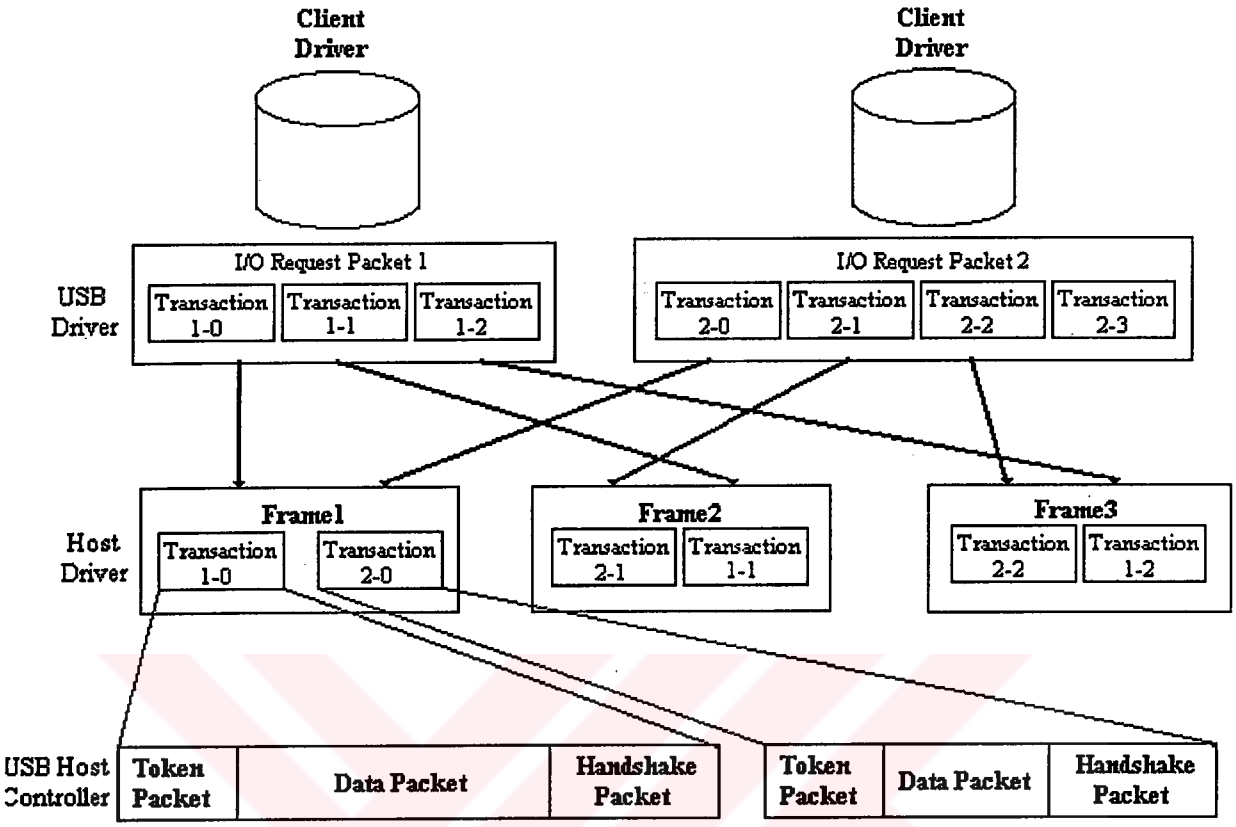
Hazırlanan listeye dayanarak host denetleyici ve kök hub USB hattı üzerinden iletilecek işlem birimlerini hazırlar. Input/output istek paketlerine (IRP) dayanarak oluşturulan işlem birimi listesi ve host denetleyici tarafından veri paketi olarak hazırlanması Şekil 2.8 'de görülmektedir. İşlem birimleri tipik olarak token, data, handshake paketlerinden oluşmaktadır. Bu paketlerin görevleri ve yapıları birbirinden oldukça farklıdır. Veri paketlerinin nitelikleri ve işlevi ayrıntılı olarak 6. bölümde açıklanacaktır.



Şekil 2.7 USB sistemine bağlanan örnek cihaz topluluğu (Anderson,1999)

2.2.3 Cihaz Tanımlayıcıları

Host, USB cihazı ile iletime başladığında yani konfigurasyon aşamasında cihazın özellikleri, cihazı oluşturan arayüzeylerin (Intrerface) ve uç noktaların (endpoint) karakteristikleri cihaz tanımlayıcıları yolu ile USB sistem yazılıma bildirilir. Ayrıca bu tanımlayıcılara ek olarak sistemde string tanımlayıcıları opsiyonel olarak da kullanılabilir. Bu tanımlayıcılar ile konfigürasyon ve arayüzey tanımları kullanıcının okuyabileceği koda çevrilir.



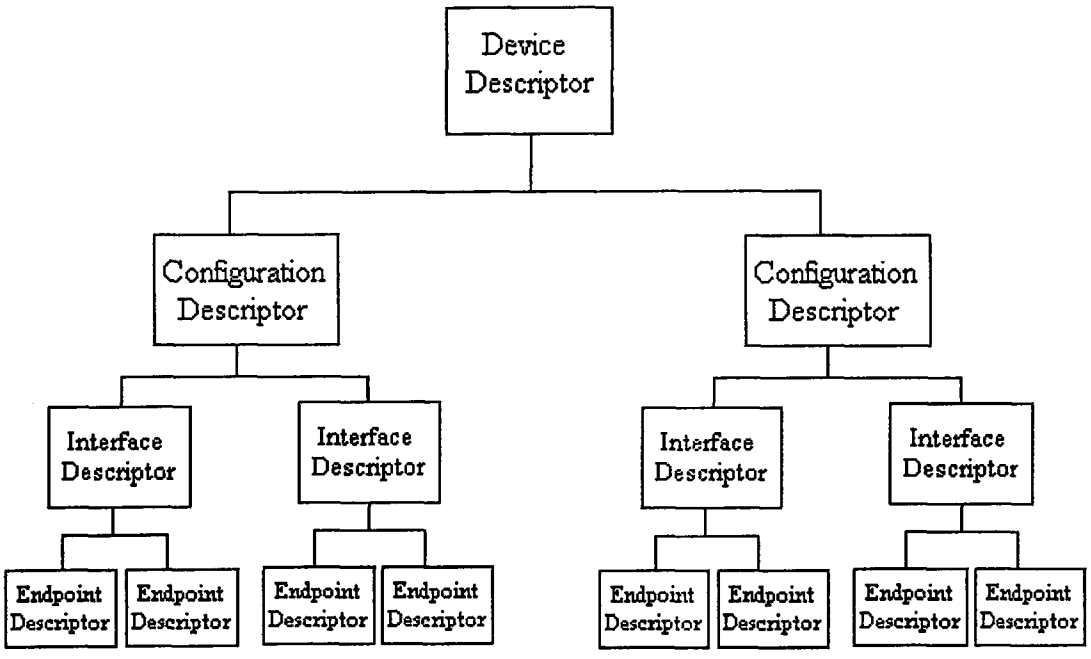
Şekil 2.8 İşlem birimi (Transaction), veri paketi ve çerçeve kavramları arasındaki ilişki (Compaq,1998)

USB cihazlarının kullandıkları ortak yöntemleri ve arabirimleri bir ortamda toplamak için çeşitli sınıflar ortaya konulmuştur. Bu sınıflara ait spesifik bilgileri hosta bildirmek için ise sınıflara özel tanımlayıcılar kullanılır. Bu sınıflara 'İnsan Arabirim Cihazı' (HID : Human Interface Device) örnek verilebilir. Bu sınıf fareler, klavyeler ve oyun kumanda cihazlarına ait ortak özellikleri gruplar. Monitor sınıfı ise görüntü pozisyonu, boyut, yerleşim bilgileri gibi monitörlerin ortak özelliklerini bir araya toplar.

İlgili cihazın karakteristikleri aşağıda listelenmiş tanımlayıcılar yolu ile host'a bildirir.

Şekil 2.9 'da standart tanımlayıcılar arasındaki hiyerarşi gösterilmiştir. Tanımlayıcılar hakkında genel bilgi 6. bölümde ayrıntılı olarak verilecektir.

- Cihaz Tanımlayıcıları
- Konfigürasyon Tanımlayıcıları
- Arayüzey (Interface) Tanımlayıcıları
- Uç nokta (endpoint) Tanımlayıcıları
- String Tanımlayıcılar
- Sınıfa özgü (Class-specific) Tanımlayıcılar



Şekil 2.9 Standart cihaz tanımlayıcıları arasındaki hiyerarşik düzen (Compaq,1998)

2.3 USB Sistem Katmanları

USB sistemi, donanım ile yazılım arasındaki ilişkiyi ve cihazın sistemle ilişkisini açıklayan 3 lojik katmandan oluşur. Şekil 2.10'da farklı katmanlar arasındaki fiziksel ve mantıksal iletişim gösterilmiştir. Katmanlı yaklaşım, sistemde bulunan ana bilgisayarın ve USB cihazın yazılım bölümleri arasındaki ilişkiyi ve bu yazılımların üstlendikleri görevleri açıklar. Bu ayrık katmanlar USB haberleşme mekanizmasını daha rahat anlamamızı sağlar. Sistemin host tarafında bulunan yazılım ve donanım elemanlarından daha önce bahsedildiğinden burada bu elemanların görevleri ve nitelikleri tekrar açıklanmayacaktır.

2.3.1 USB Bus Arayüzey Katmanı

Bu katman, host ile USB cihaz arasındaki kablo üzerinden gerçekleşen gerçek veri iletimini sağlar.

Bu katman;

- Fiziksel Bağlantı
- Elektriksel işaretleşme ortamı
- Paket iletimi mekanizmaları

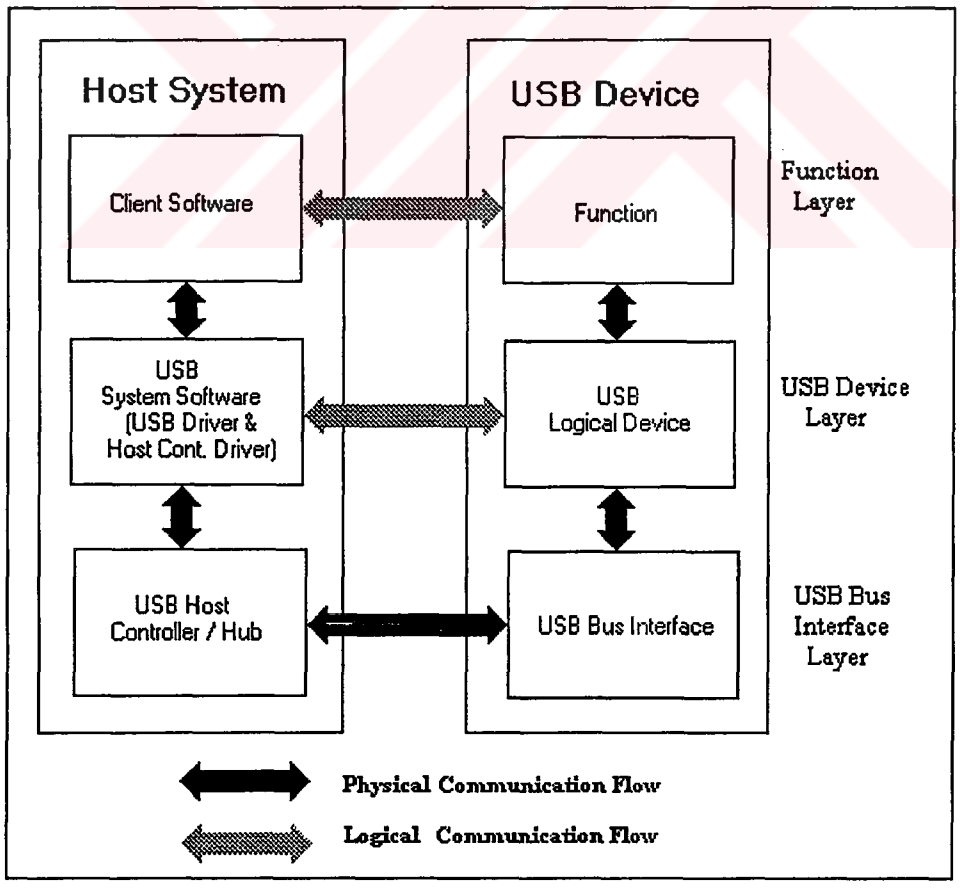
öğelerinden oluşur.

2.3.2 USB Cihaz Katmanı

Bu katman, sistemin USB cihaz tarafında yer alan bölümünün gerçek haberleşme mekanizmasını kapsar. USB sistem yazılımı lojik cihazı, cihaz üzerinde yer alan uç noktaların toplamı olarak görür. USB cihaz yazılımı, USB fonksiyonu ile bağlantı kurabilmesi için gerekli hizmetler USB sistem yazılımı tarafından sağlanır. USB sistem yazılımı cihaz tanımlayıcıları vasıtasıyla uç noktaların karakteristikleri hakkında bilgi alır. Bu bilgiler doğrultusunda USB cihaz sürücüsü, fonksiyonu ile nasıl haberleşmesi gerektiğini bilir. USB sistem yazılımı , USB bus'a bağlı tüm cihazlara erişebilir.

2.3.3 Fonksiyon Katmanı

Bu katman USB cihaz yazılımı ile ilgili cihazın fonksiyonel arayüzey arasındaki ilişkiyi açıklar. ISA , PCI'da olduğu gibi sistem fonksiyonlarına doğrudan erişemez çünkü bu fonksiyonlar doğrudan hafızada veya I/O adres boşluğunda yer almazlar. Bunun yerine USB cihaz sürücüleri cihazlara erişim yazılım arabirimini kullanır .

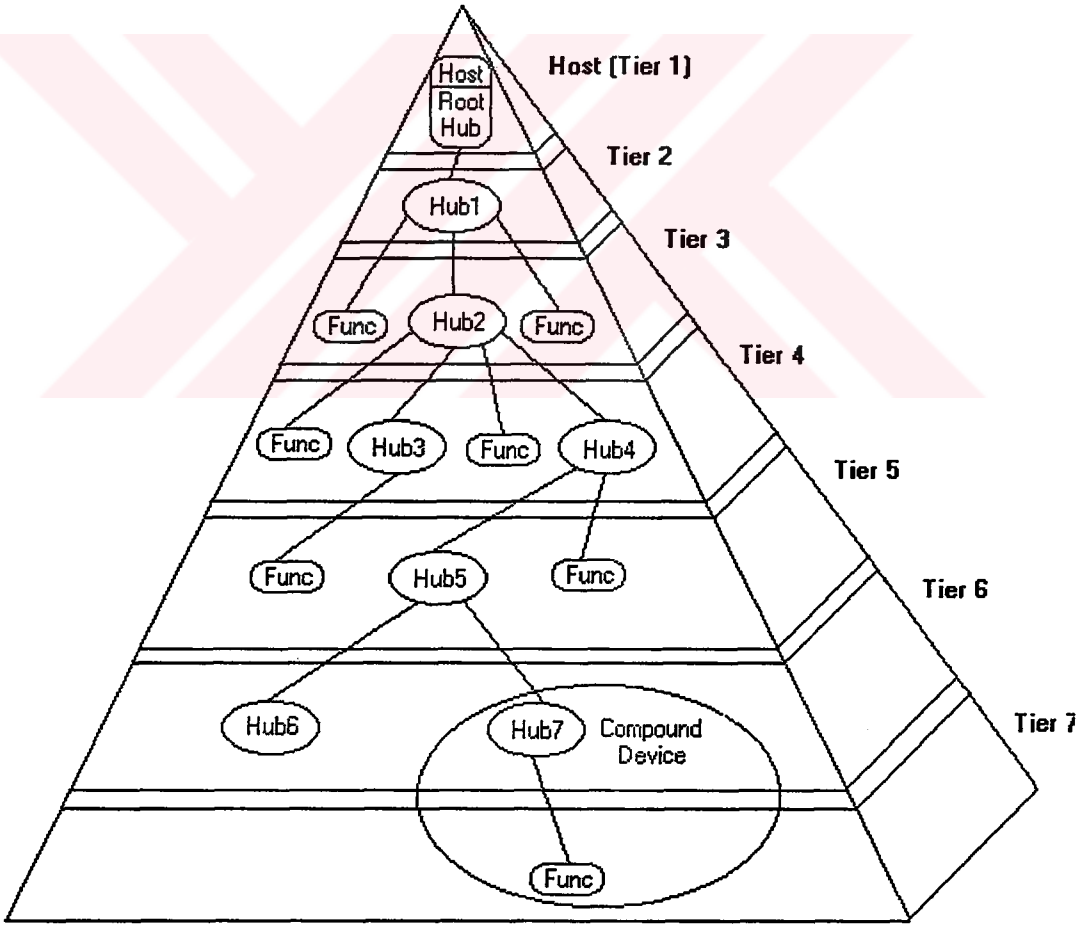


Şekil 2.10 Katmanlı yaklaşımda sistem içindeki fiziksel ve mantıksal haberleşme akışı (Anderson,1999)

2.4 USB Bus Topolojisi

Burada USB sistem içinde yer alan host, kök hub, hub ve cihazların (fonksiyon) fiziksel bağlantıları ve ağ topolojileri incelenecektir.

Bilgisayara bağlanan çevresel birimler, USB hattın band genişliğini host'un kontrolünde paylaştığından, aynı anda birçok cihaz tek hat üzerinden veri iletimini sağlayabilir. USB cihazların birbirleriyle ve host ile fiziksel bağlantısı USB hub aracılığıyla USB kablo üzerinden sağlanır. En genel haliyle sistem host, kök hub, hub ve sisteme fonksiyon sağlayan USB cihazdan oluşur. USB sistem mimarisindeki bağlantı topolojisi Şekil 2.11'de görülmektedir. Hub ve çevresel cihazlar için belirlenen zaman aşımı sınırlamasından ve kablolardaki yayılım (propagation) gecikmesinden dolayı bus topolojisi 7 kat bağlantı ile sınırlandırılmıştır.



Şekil 2.11 USB Bus Topolojisi (Compaq,1998)

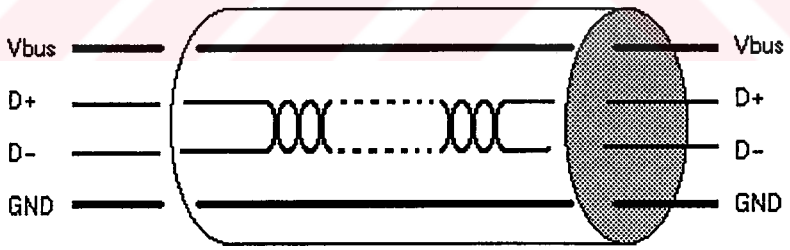
PC Host, seri hat üzerinde tüm veri alışverişini başlatan, veri transferinde kontrolü sağlayan USB sistemindeki veri akışının yöneticisidir. Bundan dolayı USB mimarisinde host efendi (master), çevresel cihazlar ise köle (slave) olarak davranır. Bus topolojisinden de görüldüğü gibi bir USB sisteminde sadece 1 tane host mevcuttur. USB Hub kullanarak bağlantı noktalarının genişletilmesi ile en fazla 127 çevresel cihaz bir host'un kontrolünde USB kablo arabirimi kullanarak veri iletimini sağlayabilir.

2.5 USB Kablo ve Konektör Yapısı

USB kablo cihaz ile host arasındaki veri iletimini gerçekleştiren fiziksel arabirimdir. Kabloların uç birimi USB hub portlarına uygun biçimde tasarlanmıştır. Kablolarda bulunan 4 farklı bağlantı teli ile cihaza gerilim aktarmak ve veri alış verişi sağlamak mümkündür.

D+ ve D- bağlantı telleri ile elektriksel bakımdan diferansiyel sinyal transferi sağlanır. Vbus ve GND bağlantı telleri ile cihaza elektriksel gerilim seviyeleri taşınır. Veri akışında arabirim olarak kullanılan USB kablo şekil 2.12'de gösterilmiştir.

Pratikte kablo içinde yer alan her bağlantı teline dışarıdan kolay tanınması açısından farklı renkler atanmıştır. Vbus : kırmızı , Gnd : Siyah , D+ : yeşil , D- : beyaz

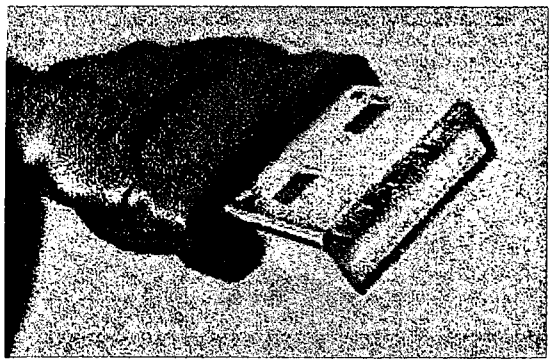


Şekil 2.12 USB Kablo (Compaq,1998)

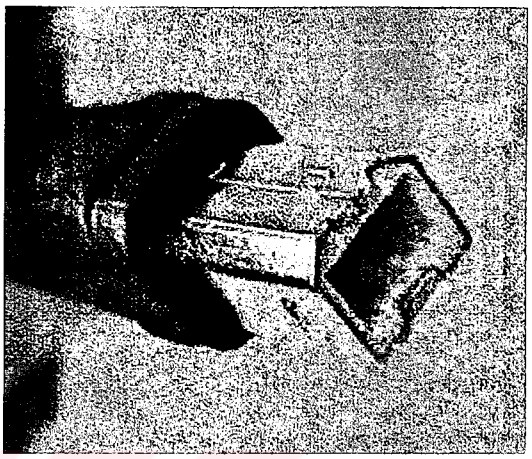
USB kablolar birkaç metre uzunlukta olabilir. Düşük hızlı (Low speed : 1.5 Mb/s) kablolar için maksimum kablo uzunluğu 3m, orta hızlı (Full speed : 12 Mb/s) kablolar için ise en fazla uzunluk 5m olmalıdır. 12Mbps'lik hızlı USB bağlantısında kullanılan kablolar iyi korunmuştur (İletim yapan kabloların etrafı hasır şeklinde ince kablo ağı ile kaplanır, bu işaretlerin elektriksel gürültüden arınmasını sağlar.). 1.5Mbps'lik yavaş USB ise daha ucuz ve nispeten korumasız bir kablo kullanır.

USB kablosunun kök hub'a ve USB cihaza (veya USB hub) bağlantısı konektör vasıtasıyla sağlanır. Cihazdan host'a doğru veri akış yönünde (yukarı veri akışı: upstream) A tipi

konektör, host'tan cihaza doğru veri akış yönünde (aşağı veri akışı : downstream) B tipi konektör kullanılır. A Tipi USB konektör Şekil 2.13'de ve B tipi USB konektör Şekil 2.14'de gösterilmiştir.



Şekil 2.13 A Tipi USB konektör

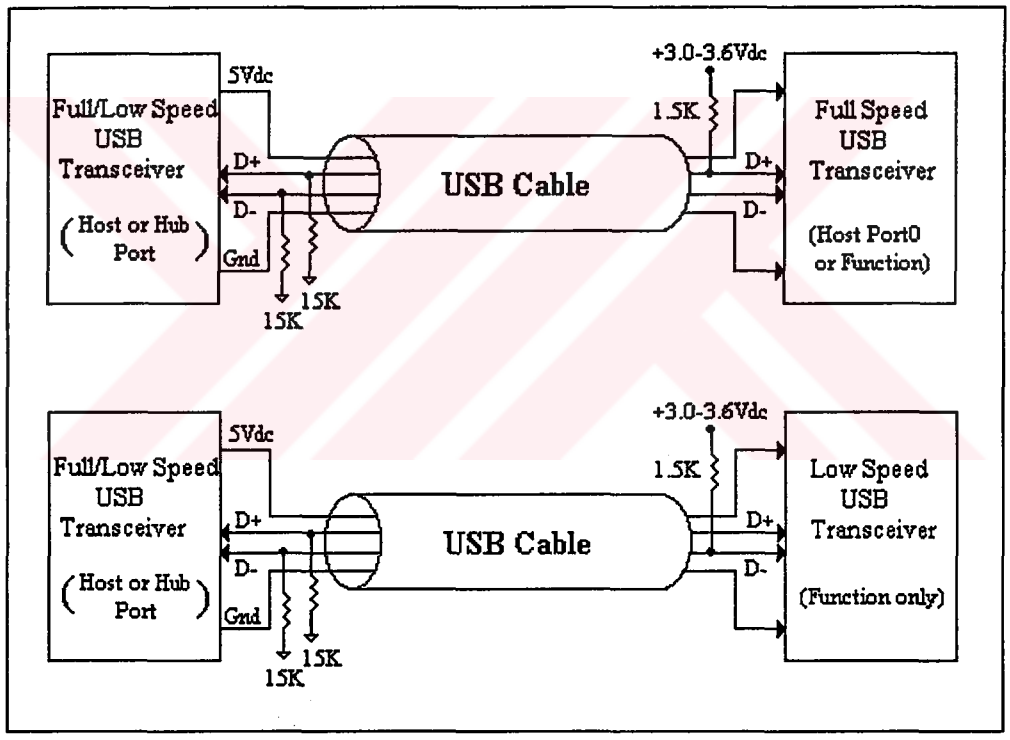


Şekil 2.14 B Tipi USB konektör

3. ELEKTRİKSEL İŞARETLEŞME

Bu bölümde, USB hat üzerinden gerçekleşen elektriksel sinyalleşme ve bu sinyallere alıcı ve verici bloğunda uygulanan işlemler incelenecektir.

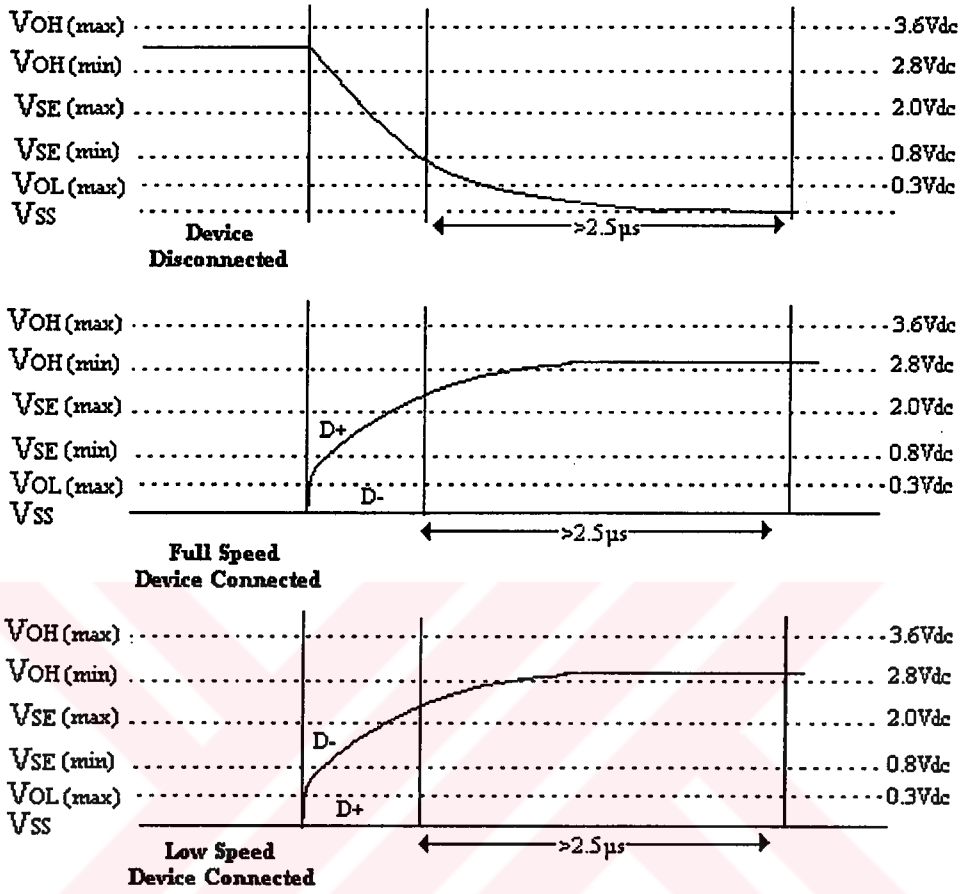
USB cihaza veri iletiminin sağlanabilmesi için öncelikle cihazın USB hattına bağlı olması gerekir. USB hub, portlarından birine USB cihazın bağlı olup olmadığını diferansiyel veri hatlarının gerilim seviyelerinin değişiminden algılar. USB portuna herhangi bir cihaz bağlı değilse D+ ve D- hatlarına bağlı olan pull-down dirençler üzerine gerilim düşümü olmaz. Her iki hat yaklaşık toprak seviyesindedir. USB cihazlar hızlarına bağlı olarak D+ veya D- hatlarına bağlı olan pull-up dirençlere sahiptirler. Alıcı ve verici bloklarına bağlı diferansiyel hatlar ve hatlara bağlı olan pull-up ve ya pull-down dirençler Şekil 3.1'de görülmektedir.



Şekil 3.1 Pull-up ve Pull-down dirençler (Anderson,1999)

Cihaz hatta bağlandığı zaman, gerilim bölümü yaratan hub'ın pull-down dirençleri ve cihazın pull-up dirençleri üzerinden akım akar. Hub, veri hatlarından birinin yaklaşık Vcc geriliminde değer aldığını diğer hattın ise yaklaşık toprak seviyesinde olduğunu tespit ettiğinde USB portuna bir cihaz bağlanmış olduğunu anlar. Aynı zamanda bağlı olan cihazın hızı da hatlardaki gerilim seviyelerinden belirlenir. Orta hızlı cihazlar D+ hattı üzerinde, düşük hızlı cihazlar ise D- hattı üzerinde pull-up dirençlere sahiptir. USB cihazın sisteme

bağlanma ve sistemden ayrılma sırasında meydana gelen hatlardaki gerilim seviyelerindeki değişim Şekil 3.2'de görülmektedir.



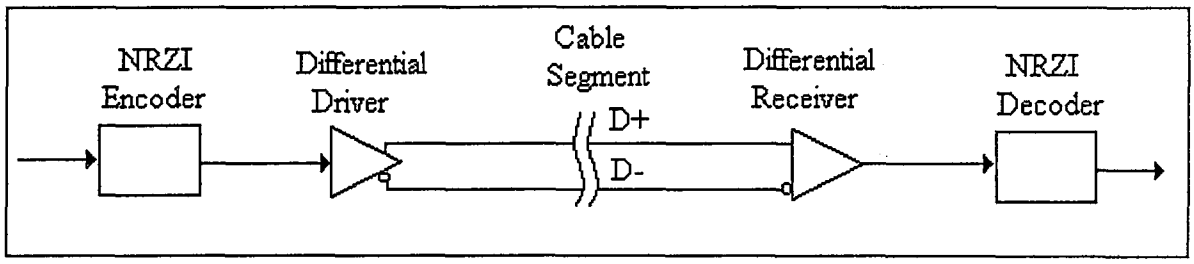
Şekil 3.2 Cihazın davranışına göre hatlardaki gerilim seviyeleri değişimi

D+ ve D- hatlarındaki her iki sinyal Vse(min)=0.8Vdc gerilimine düştüğünde USB hub, cihazın sistemden ayrıldığını anlar. D+ ve ya D- hatlarındaki Vse(max)=2Vdc seviyeye ulaştığında ise USB hub cihazın sisteme bağlandığını tespit eder. Cihaz sistemden ayrıldığında hub, port durum yazmacında bulunan durum bitlerini resetler, cihaz sisteme bağlandığında ise bu bitleri set eder. Host yazılımı herhangi bir cihazın sisteme bağlı olmadığını sistemdeki tüm hub'ların portlarını periyodik olarak kontrol ederek algılar.

3.1 NRZI Kodlama İşlemi

USB seri veri USB hattına çıkarılmadan önce NRZI (Non Return to Zero Inverted) kodlama işleminden geçirilir. Bu işlemin sonucunda elde edilen kodlanmış veri, diferansiyel sürücü tarafından USB kabloya çıkarılır. Alıcı tarafında, gelen seri diferansiyel işaret kuvvetlendirilir ve NRZI kod çözücüden geçirilir. Kodlama ve diferansiyel işaretleşme verinin doğru ve

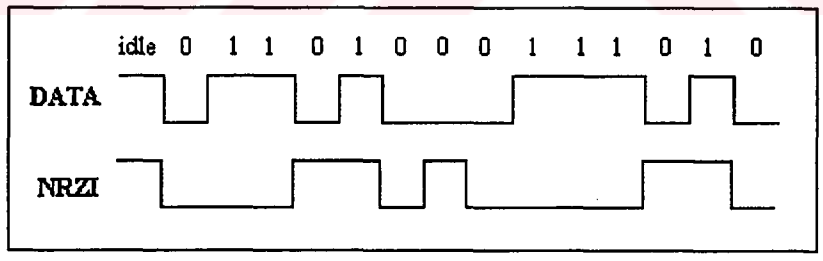
gürültü problemlerinden arındırılmış olarak iletimini garanti eder. En genel haliyle alıcı ve verici tarafta kodlama , kod çözme ve diferansiyel işaretleşme Şekil 3.3'te gösterilmiştir.



Şekil 3.3 NRZI kolama ve NRZI kod çözücü bloklarının alıcı ve verici tarafta gösterimi

NRZI kodlama işleminin farklı veri iletim sistemlerinde uygulamasını görmek mümkündür. Seri veri içinde yer alan '1' lojik seviyeleri çıkış seviyesinde bir değişiklik yaratmamasına rağmen, '0' lojik seviyesi çıkış seviyesinin değişmesine yol açar. NRZI kodlayıcı veriyi doğru olarak örnekleyebilmek için iletilecek veri ile senkronizasyonu sürdürmek zorundadır. Arka arkaya '1' lojik seviyesinin algılanması, NRZI kodlanmış verinin uzun bir süre konum değiştirmeden sabit kalmasına yol açar. Bundan dolayı veri ile NRZI kodlayıcı arasındaki senkronizasyon bir süre sonra kaybolur. Bunu engellemek üzere NRZI kodlama işleminden önce veri bit stuff işleminden geçirilir.

Şekil 3.3'te iletilecek seri veri ve bu veriye uygulanan NRZI kodlama işlemi gösterilmiştir

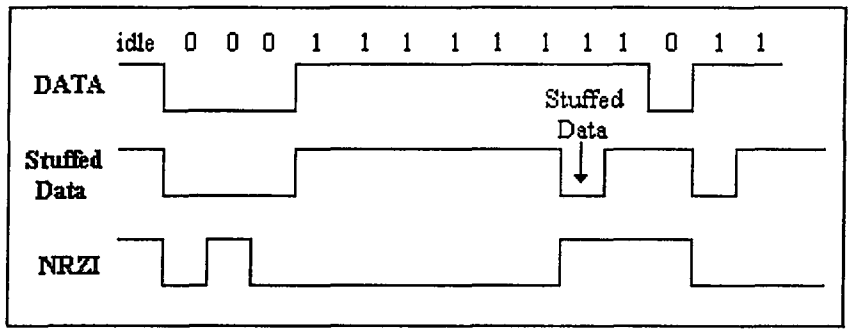


Şekil 3.4 Seri veri ve bu veriye uygulanan NRZI kodlama işlemi

3.2 Bit Stuff İşlemi

Gönderilen seri veri, arka arkaya altı bit lojik '1' seviyesinde ise araya lojik '0' seviyesi eklenerek gelen veri ile alıcı tarafın senkronizasyonu sağlanır. NRZI kodlama işleminde, seri veride her lojik '0' seviyesi gördüğünde bir geçiş elde edildiğinden, ardışıl altı lojik '1' seviyesinden sonra gelen lojik '0' seviyesi NRZI kodlanmış veride bir geçiş olmasını sağlayacaktır. Şekil 3.5'te USB hatta çıkarılacak seri verinin önce bit stuff ve ardından NRZI kodlama işleminden geçirilmesi gösterilmiştir. Bit stuff işlemi ile altıncı ile yedinci bitler

arasına lojik '0' seviyesi eklendiğinden, yedinci lojik '1' değeri bir veri süresi kadar geciktirilmiştir. Alıcı taraf altı lojik '1' seviyesi ardından gelen lojik '0' değerinin bit stuff nedeniyle eklenen seviye olduğunu bilir ve bu bir bitlik veri bit de_stuff işlemi sonucu ayıklanır. Eğer alıcı, veri paketi içerisinde herhangi bir yerde ard arda yedi adet lojik '1' değeri görürse bit stuff hatası meydana gelir ve veri alma işlemi yarıda bırakılır.



Şekil 3.5 Seri verinin önce bit stuff ve ardından NRZI kodlama işleminden geçirilmesi

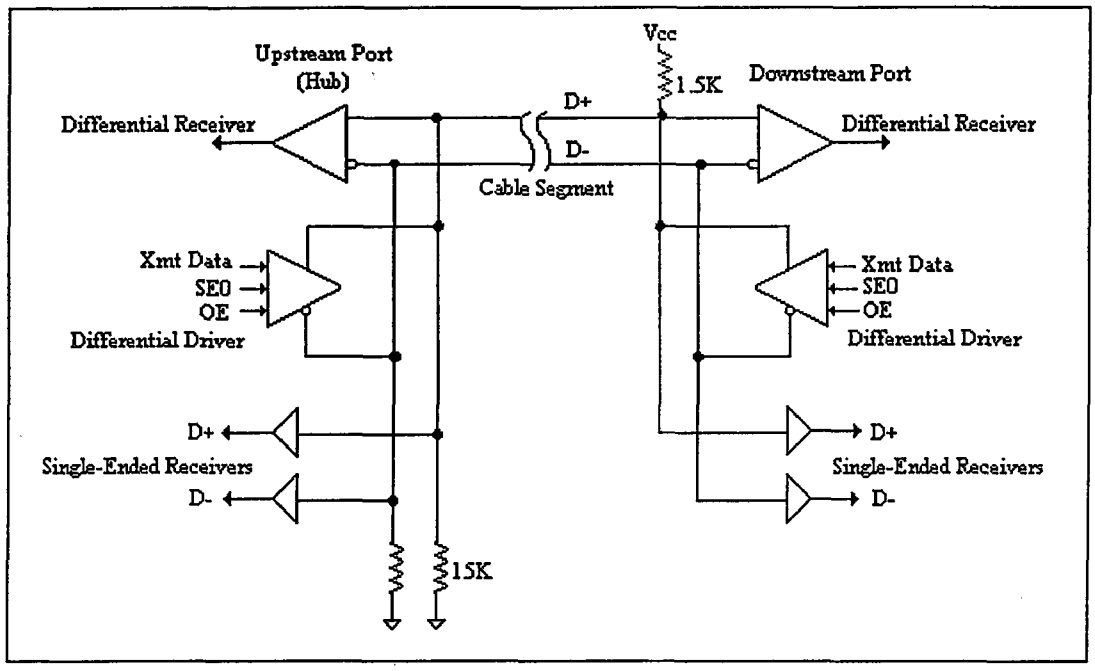
3.3 Diferansiyel İşaret Çifti

İşaretteki gürültüyü azaltmak için USB hattı üzerinde diferansiyel sinyalleşme kullanılır. Sürücü ve alıcı tarafta işarete uygulanan kuvvetlendirme işlemi sırasında ve/veya kablo üzerinden iletim sırasında elektromagnetik alandan dolayı oluşan gürültü işaretin bozulmasına neden olabilir.

İletim esnasında verinin bozulmasına neden olan gürültüyü azaltmak, gelen sinyali kuvvetlendirmek ve hattın durumunu algılayabilmek üzere USB hub ve cihaz tarafında iletim hattıyla arayüz oluşturan bloklar tanımlanmıştır. Bu bloklar;

- Diferansiyel Alıcı Bloğu
- Diferansiyel Sürücü Bloğu
- Single Ended Alıcı Bloğu

USB hub ve cihaz tarafında iletim hattıyla arayüz oluşturan bloklar Şekil 3.6'da gösterilmiştir.

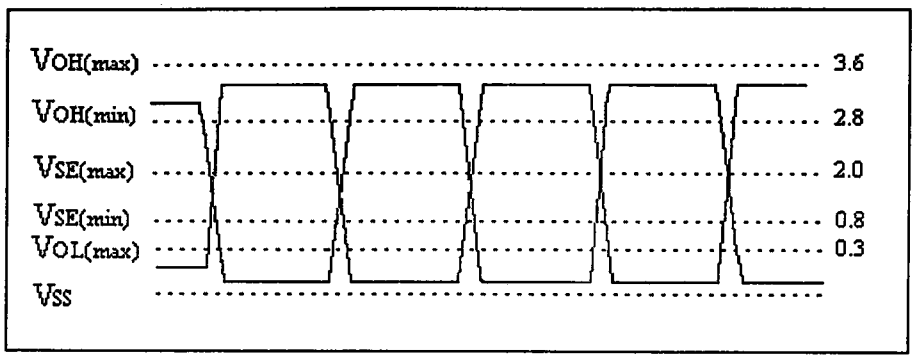


Şekil 3.6 USB hub ve cihaz tarafında iletim hattıyla ara yüzey oluşturan bloklar (Anderson,1999)

Veri gönderen tarafın diferansiyel sürücü çıkışında elde edilen diferansiyel işaret D+ ve D- iletim hatları üzerinden taşınır. Alıcı tarafta bu işaret diferansiyel alıcı bloğu tarafından alınarak farksal kuvvetlendiriciden geçirilir. Single ended alıcı bloğu ise D+ ve D- hatlarıdaki gerilim seviyelerini dikkate alarak USB hattın durumunu tespit eder. Örneğin diferansiyel hatlar 2.5μs'den uzun bir süre lojik '0' seviyesinde ise single ended alıcı bloğu hatta RESET sinyalinin gönderildiğini algılamaktadır.

USB half duplex modda çalıştığından aynı anda karşılıklı veri akışı söz konusu değildir. Alıcı ve gönderici kavramları iletimin yönüne bağlı olarak cihaz ve ya hub (dolayısıyla host) sistem elemanlarının yerine geçer.

USB hat üzerinden gönderilen veri ile birlikte senkronizasyonu sağlayan saat (clock) sinyali iletilmediğinden, hat üzerinde asenkron haberleşme akışı vardır. USB cihazlar kendi saat işaretlerine sahip olduklarından (orta hızlı : 48 MHz , düşük hızlı : 6 MHz) cihazın saat girişi ile gelen verinin senkronizasyonu alınan verinin ilk bitleri (SYNC) ile sağlanır. Şekil 3.7'de USB hat üzerinden iletilen diferansiyel işaret gösterilmiştir. Lojik '1' değerinin algılanacağı maksimum gerilim seviyesi 3.3Vdc 'dir.



Şekil 3.7 USB hat üzerinden iletilen diferansiyel işaret

4. İLETİM TİPLERİ

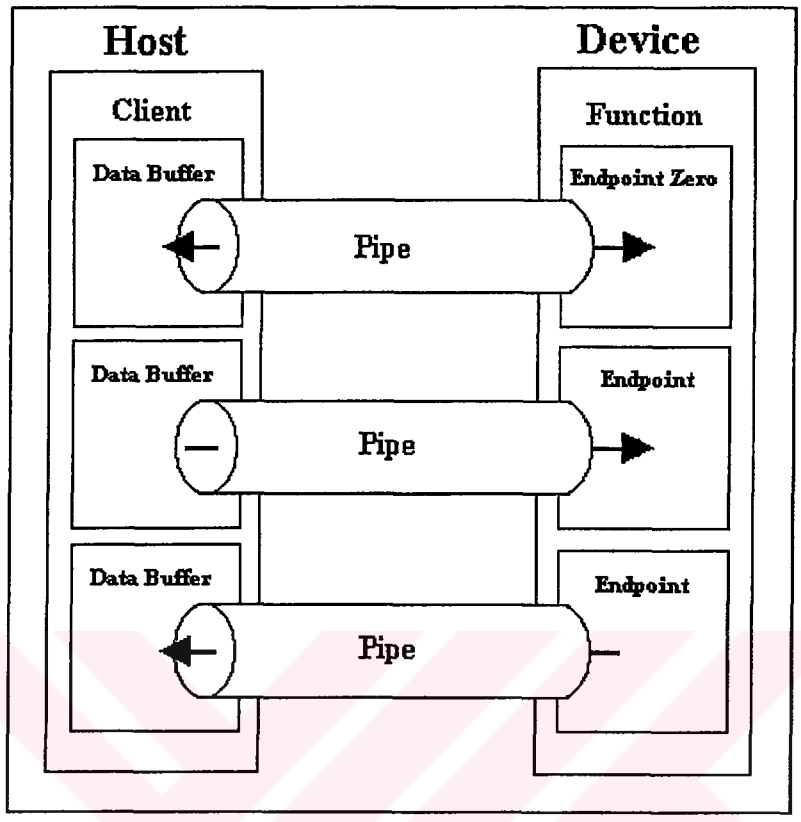
Bu bölümde, USB cihazların gereksinimlerine bağlı olarak tanımlanan farklı veri iletim tipleri incelenecektir. İlk olarak host tarafında bulunan yazmaçlar ile cihaz tarafında bulunan uç noktalar arasında gerçekleşen haberleşme kanalları hakkında bilgi verilecektir.

4.1 Haberleşme Kanalları ve Uç noktalar (Endpointler)

USB haberleşme sisteminde host ile USB cihazın servis ihtiyaçlarını karşılamak ve farklı veri yapılarını idare etmek amacıyla 4 farklı transfer tipi tanımlanır. İletim tiplerinin karakteristikleri, ilgili transfere ait uç nokta (endpoint) tanımlayıcıları vasıtasıyla belirlenir. Her transfer tipi bir veya daha fazla uç noktaya karşı düşer. Cihaz, USB sistemine uç noktaların toplamı olarak görünür ve uç noktaların oluşturduğu küme interface olarak adlandırılır. Bir cihaz içinde birden fazla interface olabilir. USB cihaz sürücüsü, cihaz içindeki herhangi bir interface'i haberleşme kanalları vasıtası ile idare eder. Cihazda bulunan uç nokta (endpoint) ile host tarafında bulunan yazmaçlar (client registers) arasında oluşan haberleşme kanalı, USB cihaz sürücüsü transfer talebinde bulunduğu sürece aktif kalır. Şekil 4.1'de host tarafında bulunan yazmaçlar ile cihaz tarafında bulunan uç noktalar arasında gerçekleşen haberleşme kanalları gösterilmiştir.

Konfigürasyon işlemi sırasında, USB sürücüsü, cihaz tanımlayıcılarında bulunan bilgileri okuyarak fonksiyon için gerekli olan uç nokta sayısını ve iletim tipini belirler. Ayrıca bu bilgilerden yararlanılarak her uç nokta için gerekli olan maksimum band genişliğini tespit eder. Host tarafından ilgili uç nokta için gerekli band genişliği sağlanıyorsa uç nokta ile host (client) arasında haberleşme kanalı USB sürücüsü tarafından oluşturulur. 3. bölümde ayrıntılı olarak bahsedildiği gibi USB cihaz sürücüsü, uç nokta tanımlayıcılarından aldığı bilgiler doğrultusunda transfer talebinde bulunur. Maksimum band genişliklerine bağlı olarak her 1ms'lik çerçeve (frame), belirli transfer tiplerine USB sürücü tarafından ayrılır. Verinin boyutuna bağlı olarak her transfer birkaç çerçevede tamamlanabilir. USB sürücü tarafından sunulan transfer talepleri (IRP), host denetleyici sürücüsü tarafından uç nokta tanımlayıcılarından alınan bilgiler doğrultusunda USB hatta çıkarılmak üzere listelenir. Host denetleyici ise listedeki transfer taleplerini veri paketi bazında oluşturur. Transfer talebinin yönüne bağlı olarak veri iletimi cihazdan host'a veya host'tan cihaza doğru olabilir. USB hattı çeşitli cihazlar ve bu cihazlara ait uç noktalar arasında paylaşılır. Bir çerçevenin %10 'u periyodik olmayan transferler tarafından, %90'ı ise periyodik transferler tarafından paylaşılır. Transfer tiplerini ayrıntılı incelemeden önce uç nokta tanımlayıcılarında yer alan bilgiler

tanıtılacaktır.



Şekil 4.1 Haberleşme Kanalları (Anderson, 1999)

4.2 Transfer Tipleri

İletim için gerekli olan transferlerin karakteristikleri uç nokta tanımlayıcıları yolu ile host'a bildirilir. Konfigürasyon işleminden bahsedilen 6. Bölümde, bu tanımlayıcılar ayrıntılı olarak listelenecektir. Her uç nokta kendi özelliklerini belirten 7 byte'lık tanımlayıcılara sahiptir. Tanımlayıcıda bulunan ilgili uç noktaya ait adres, yön (IN ve ya OUT), transfer tipi (Kontrol, Yığın (Bulk), Interrupt (Kesme), Isochronous), maksimum paket uzunluğu, periyodik transferler için kontrol aralığı gibi özellikler konfigürasyon sırasında host'a iletilir. Bu özelliklere transfer tipine bağlı olarak bazı sınırlamalar getirilmiştir. USB 1.1 protokolü düşük hızlı cihazlar için kesme ve kontrol transferlerini destekler, orta hızlı cihazlar için ise uygulamada tüm transfer tipleri geçerlidir.

Tüm transfer tipleri için oluşturulan veri paketleri aynı yapıdadır. Zaman ve nitelik kavramlarındaki doğruluk faktörü her transfer tipi için farklı öneme sahiptir. Çizelge 4.1'de her transfer tipi için bu özellikler değerlendirilmiştir.

Çizelge 4.1 Transfer Tipleri

Transfer tipleri	Önemli Özellikler	Örnek Cihazlar
Kesme (Interrupt)	Nitelik	Fare (Mouse), Klavye
Yığın (Bulk)	Nitelik	Yazıcı , Tarayıcı
Isochronous	Zaman	Hoperlör , Video
Kontrol	Zaman , Nitelik	Sistem Kontrol

İletim zamanındaki doğruluk host tarafından sağlanır. Her çerçevenin belli bir kısmı zaman faktörünün önemli olduğu belirli transfer tipleri için ayrılır. Bu işlem zamanda doğruluk faktörünü garanti eder.

İletimde nitelik yani verinin iletim sırasında bozulmaması ise hata kontrolü, hata düzeltme ve el sıkışma (handshake) mekanizması kullanılarak garanti altına alınır.

Uygulamadaki gereksinimlere bağlı olarak dört farklı transfer tipi tanımlanır. Takip eden bölümde kesme (interrupt), yığın (bulk), isochronous, kontrol transferlerin özellikleri açıklanmıştır.

4.2.1 Kesme (Interrupt) Transfer Tipi

Sisteme bağlanan cihazdan dış etkenler sonucu meydana gelen ani reaksiyonlara cevap verebilmek amacıyla host, veri transferini başlatır. Mikroişlemcilerde kullanılan kesme yapısından farklı olarak USB sistemine dışarıdan veri talebi olup olmadığı periyodik olarak host yazılımı tarafından sorgulanır. USB sisteminin kesme transfer tipini destekleyen cihazı kontrol etme aralığı, cihazın hızına göre belirlenir. Host bu işlemin periyodunu, konfigürasyon sırasında cihaz tanımlayıcılardan okuduğu bilgiler doğrultusunda tespit eder. Cihaz tarafından belirlenen frekans değeri veriyi kaybetmeyecek kadar büyük, gereksiz yere hattı meşgul etmeyecek kadar küçük bir değere sahip olmalıdır.

4.2.1.1 Kesme Transfer Tipinin Özellikleri

- Orta hızlı (Full-speed) ve düşük hızlı (Low-speed) cihazlar tarafından desteklenir.
- Periyodik transferdir.
- Maksimum veri uzunluğu orta hızlı iletim için 64 byte veya daha az, düşük hızlı iletim için 8 byte veya daha azdır.
- Veri paketlerinin maksimum uzunlukta gönderilmesi gerekmez.(Tüm transfer tipleri için

geçerlidir.)
- Veri paketi maksimum uzunluktan daha az ise maksimum uzunluğa şişirilmez. (Tüm transfer tipleri için geçerlidir.)
- Her 1ms'lik çerçevenin en fazla %90 band genişliği periyodik transferler (Interrupt, ISO) için ayrılmıştır.
- USB hatta erişim periyodu ve maksimum veri uzunluğu konfigürasyon sırasında uç nokta tanımlayıcıları (= endpoint descriptor) vasıtasıyla belirlenir.
- USB hatta erişim periyotları ;

  Orta hızlı iletim için 1-255ms

  Düşük hızlı iletim için 10-255ms

  arasında cihazın gereksinimlerine göre bağlı olarak değişir.
- Hatalar belirlenen periyodun aşılmasına neden olur.
- Host ile senkronizasyonu sağlamak amacıyla kesme uç noktaları (interrupt endpoint) bir bitlik veri senkronizasyon yazmaçlarına sahiptir.
- Uç nokta yazmacının meşgul olması veya zaman aşımı durumunda (timeout) veri paketi bir sonraki periyotta tekrar gönderilir.
- Uç nokta yazmacının meşgul olması veya belirlenen periyotta veri paketinin bulunmaması durumunda host'a bu durum 1 byte'tan oluşan özel veri paketi (NAK : No Acknowledge Handshake) ile bildirilir.

4.2.2 Yığın Transfer Tipi

Zaman kritik olmayan çok büyük miktarlarda verinin iletimi için kullanılır. Veri iletimi çerçeve (frame) içinde yeterli band genişliğinin varlığına bağlıdır.

4.2.2.1 Yığın Transfer Tipinin Özellikleri

- Orta hızlı (Full speed) cihazlar tarafından desteklenir.
- Periyodik transfer değildir.
- USB hatta erişim geçerli band genişliğinin varlığına bağlıdır.
- İletimde nitelik yani verinin iletim sırasında bozulmaması garanti edilir.
- İletimde zaman faktörü önemli değildir. İletim için geçerli band genişliği sağlanana kadar veri geciktirilebilir.
- Kabul edilebilir maksimum veri uzunluğu 8, 16, 32 ve ya 64 byte'dır
- Her çerçevenin en fazla %10 band genişliği periyodik olmayan transferler (Yığın, Kontrol) için ayrılmıştır.
- Host ile senkronizasyonu sağlamak amacı ile yığın uç noktalar 1 bitlik veri senkronizasyon yazmacına sahiptir.

4.2.3 Isochronous Transfer Tipi

Uygulamada mikrofon, kamera, hoparlör gibi cihazlarda kullanıldığından gönderici ve alıcı taraftaki uyum ve zaman faktörü çok önemlidir.

4.2.3.1 Isochronous Transfer Tipinin Özellikleri

- Orta hızlı cihazlar tarafından desteklenir.
- Periyodik transferdir.

- Maksimum veri uzunluğu 1023 byte ile sınırlandırılmıştır.
- Her çerçevenin en fazla %90 band genişliği periyodik transferler (Interrupt , ISO) için ayrılmıştır.
- Her çerçevede bir ve ya daha fazla ISO veri paketi iletilir.
- Verinin çerçeve içindeki yeri garanti değildir. Zamanda hizalamayı sağlamak amacıyla gelen veri uç nokta yazmacında 1 çerçeve bekletilir.
- Zamanlama faktörü verinin kayıpsız iletilmesinden daha önemlidir.
- Zamanda gecikmenin önlenmesi amacıyla ISO transferler el sıkışma (handshake) ve hata durumunda verinin tekrar iletimi mekanizmasını desteklemezler.

4.2.4 Kontrol Transfer Tipi

USB cihaz sisteme ilk bağlandığında cihaza ve cihazda bulunan uç birimlere ait özellikler konfigürasyon sırasında kontrol transferi aracılığı ile hosta bildirilir. Konfigürasyon işlemini gerçekleştiren kontrol transfer vasıtası ile sıfırcı uç nokta olarak da adlandırılan default kontrol uç noktasına haberleşme kanalı ile erişilir. Default kontrol kanalı ile cihaza ait konfigürasyon, durum, kontrol bilgileri host tarafından okunur (ve ya yazma işlemi de sağlanabilir.). Kontrol transferi, talep edilen verinin tipine bağlı olarak 3 fazdan oluşabilir.

4.2.4.1 Setup fazı

Talep edilen verinin tipini ve hedef birimi (cihaz , interface , uç nokta) belirleyici veri yapısına sahip olan 8 byte'lık veriyi içerir. Ayrıca bir sonraki fazda (veri fazı, opsiyonel) iletilecek verinin yönünü ve uzunluğunu belirten veriye sahiptir.

4.2.4.2 Veri Fazı

Setup fazındaki iletilen veri talebine bağlı olarak üretilir. Cihazın niteliklerini betimleyen bilgiler host'a veri fazında taşınır. Ayrıca bu fazda host tarafından belirli değer atama işlemleri de (set) gerçekleşebilir.

4.2.4.3 Durum (Status) Fazı

Kontrol transferin başarılı sonuçlanıp sonuçlanmadığını her iki birime durum (status) fazı ile bildirilir. Durum fazındaki transferin yönü veri fazındaki iletim ile ters yönlüdür. Eğer veri fazı bulunmuyorsa veri iletimi daima host'a doğrudur.

4.2.4.4 Kontrol Transfer Tipinin Özellikleri

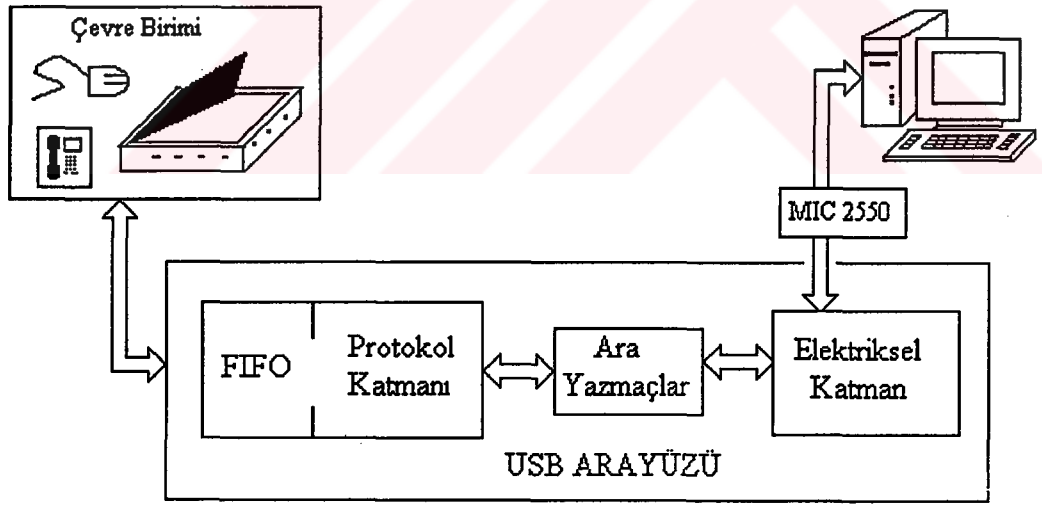
- Düşük ve orta hızlı tüm cihazlar tarafından desteklenir.
- Periyodik transfer değildir.
- İletimde nitelik yani verinin iletim sırasında bozulmaması garanti edilir.
- Hatalı verinin aynı çerçevede tekrar gönderilmesi şart değildir.

- Maksimum veri uzunluğu orta hızlı iletimde 8, 16, 32 ve ya 64 byte , düşük hızlı iletim için ise 8 byte ile sınırlandırılmıştır.
- Her çerçevenin en fazla %10 band genişliği periyodik olmayan transferler (Yığın, Kontrol) için ayrılmıştır.
- Kontrol transferi yığın transferine göre öncelik hakkına sahiptir. Kontrol transferi periyodik olmayan transferler için ayrılan band genişliğini kullanmadığı zaman, kalan kısmı yığın transferi kullanabilir.
- Host ile cihaz arasındaki senkronizasyonu sağlayan mekanizmaya sahiptir.

5. USB CİHAZ ARAYÜZÜ TASARIM KRİTERLERİ

İTU ETA ASIC tasarım merkezinde yürütülmekte olan proje kapsamında USB cihaz arayüzü tasarımı hedeflenmektedir. Bu birim, USB cihaz ile USB haberleşme hattı arasındaki dijital donanımı kapsamaktadır. Bu tasarımda USB 2.0 spesifikasyonun düşük hızlı (Low-speed) ve orta hızlı (Full-speed) kısımlarını tanımlayan USB1.1 spesifikasyonu temel alınmıştır. Haberleşmedeki yazılım birimleri ve arayüz ile USB hattı arasındaki analog işaretleşmeyi sağlayacak blokların tasarımı bu spesifikasyonun dışındadır. Tasarımda USB hat ile analog bağlaşım için MICREL MIC 2550 USB transceiver tüm devresi kullanılacaktır. Bu yüzden arabirimin host tarafı bu tüm devre ile uyumlu olacak şekilde tasarlanmıştır. USB arabiriminin cihaz ile haberleşmesi ise FIFO bellek elemanı üzerinden sağlanacaktır. Arayüz öncellikle düşük hızlı çevre birimleri haberleşecek şekilde tasarlanmış, ancak orta hızlı tanımlamalar doğrultusunda bazı standartlar desteklenmiştir.

Tasarımı hedeflenen USB cihaz arayüzünün USB sistem içinde bulunduğu konuma ilişkin, blok diyagram Şekil 5.1'de gösterilmiştir. Tasarlanmakta olan sistem iki ana bölümden oluşmaktadır.



Şekil 5.1 USB cihaz arayüzünün sistem içindeki konumu

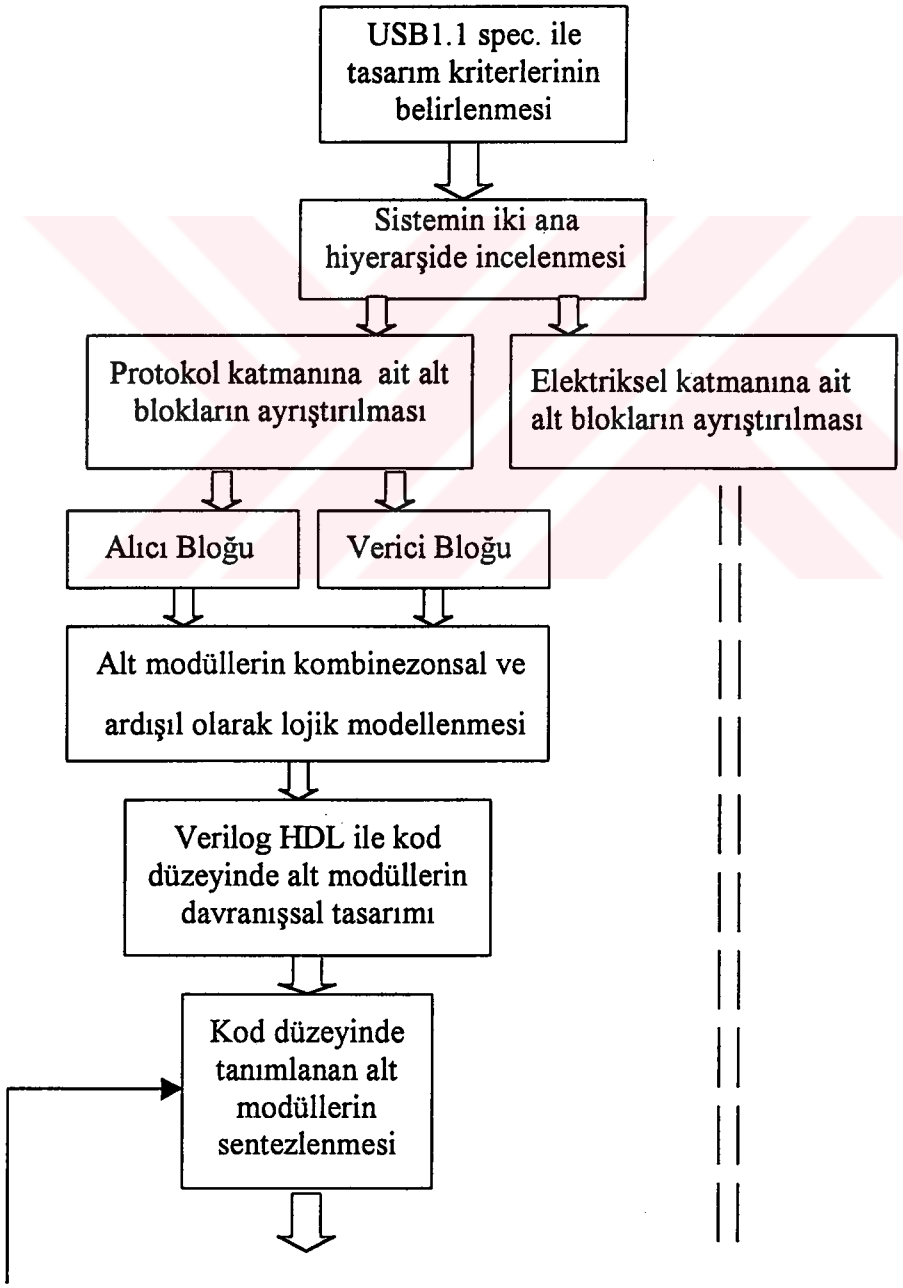
Protokol Katmanı : Çevre birimi ile bağlaşımı ve üst seviye haberleşme protokolünü sağlar. Cihaz ile haberleşme FIFO üzerinden gerçekleşir. Bu katmanı oluşturan bloklarının tasarımı ve simülasyon sonuçları 6. bölümde ayrıntılı olarak incelenecektir.

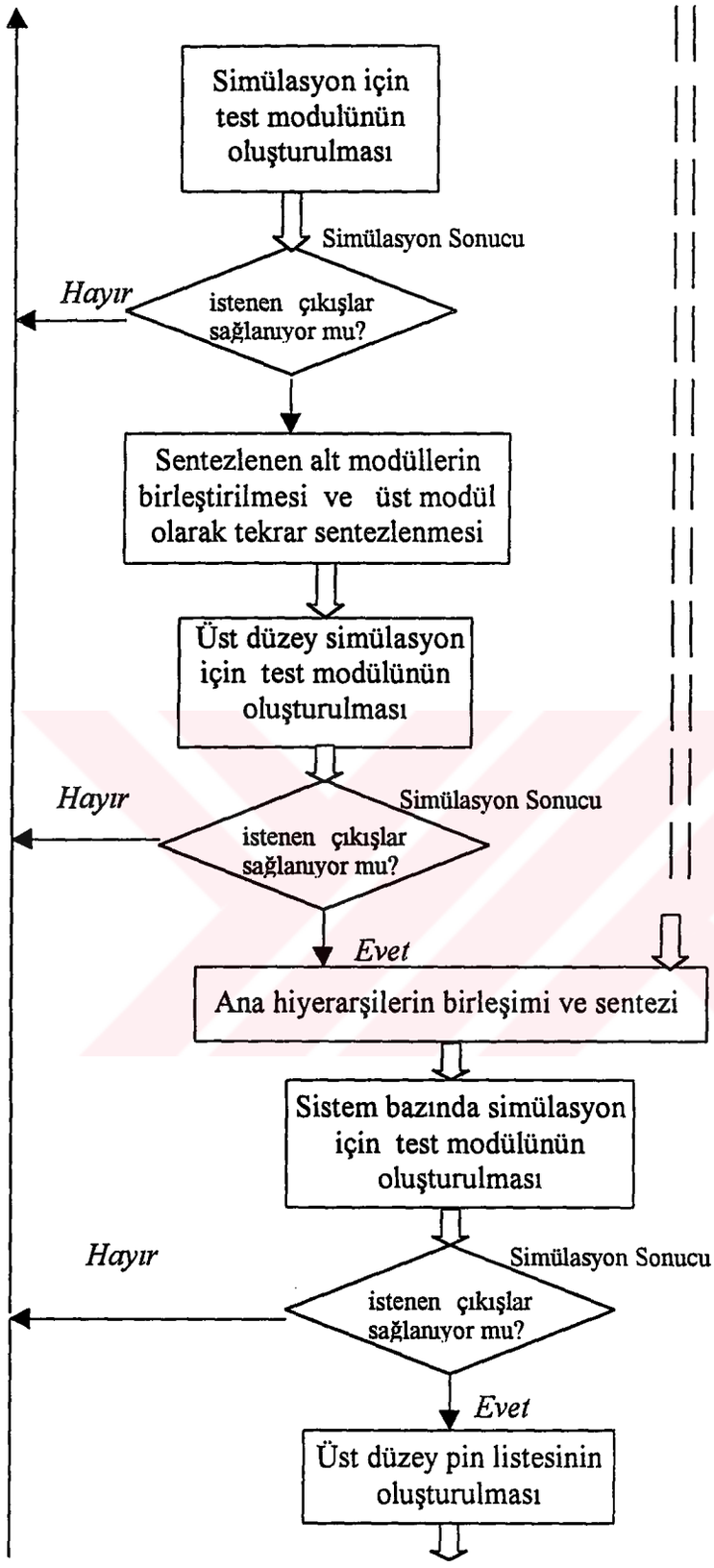
Elektriksel Katman : MICREL'in MIC 2550 USB transceiver entegresinden elde edilen dijital seri verinin işlenip, protokol katmana iletiminden sorumludur. Bu katmanın görevlerine ilişkin tanımlamalar 4. Bölümde yer alan elektriksel sinyalleşme (NRZI kodlama, bitstuff, senkronizasyon...) kısmında değinilmiştir.

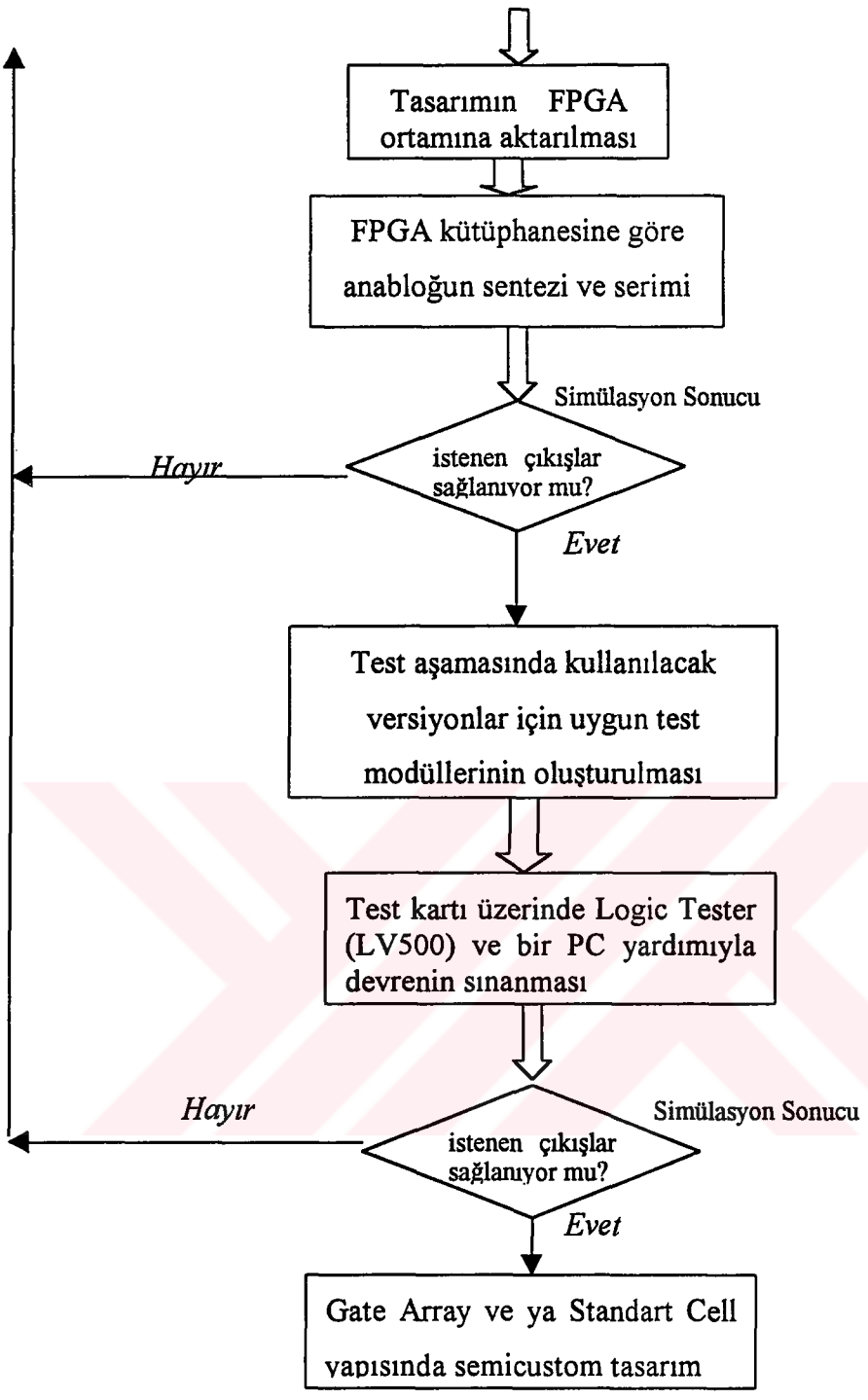
Protokol katmanı ile elektriksel katmanın birbirleri ile uyumlu bir şekilde haberleşmesi için 'Ara Yazmaçlar' adıyla Şekil 5.1'de gösterilen ara blok tasarlanmıştır. Bu blok veri alma-gönderme (receive / transmit) işlemleri için 8 bitlik yazmaç ve alma /gönderme işlemlerinin durumu gösteren, iki katman arasındaki uyumu ve kontrolü sağlayan çeşitli bayraklar içerir.

5.1 Tasarım Süreci

Burada ETA ASIC tasarım merkezinde devam etmekte olan projenin başlangıcından bu yana geçen evreler anlatılacaktır. Tasarım sürecine ilişkin akış diyagramı aşağıda verilmiştir.




Simülasyon için test modulünün oluşturulması
Simülasyon Sonucu
Hayır
istenen çıkışlar sağlanıyor mu?
Sentezlenen alt modüllerin birleştirilmesi ve üst modül olarak tekrar sentezlenmesi
Üst düzey simülasyon için test modülünün oluşturulması
Hayır
Simülasyon Sonucu
istenen çıkışlar sağlanıyor mu?
Evet
Ana hiyerarşilerin birleşimi ve sentezi
Sistem bazında simülasyon için test modülünün oluşturulması
Hayır
Simülasyon Sonucu
istenen çıkışlar sağlanıyor mu?
Evet
Üst düzey pin listesinin oluşturulması



Şekil 5.2 Gerçekleştirilmekte olan projenin tasarım sürecine ilişkin akış diagramı

Tasarım sürecinde, dijital sistem uygun alt modüllere ayrıştırılarak tanımlanmıştır. Alt modüller USB1.1 spesifikasyonunun belirlediği kriterlere uyularak kombinezonsal ve/veya ardışıl olarak blok bazında tasarlanmıştır. Verilog HDL tasarım dili kullanılarak kod düzeyinde alt modüllerin davranışsal tasarımı gerçekleştirilmiştir. Kod düzeyinde tasarlanan alt modüller Synopsys - Design Analyzer sentezleyici aracı kullanılarak sentezlenmiştir. Sentezleme adımı, kod düzeyinde tasarımın lojik kapı düzeyinde eşdeğerinin oluşturulmasını sağlar. Ayrıca sentezleme sonrası devrenin güç, frekans, alan ve gecikme gibi tasarım

parametreleri belirlenir. Alt modüllerin çıkışlarını (output) gözlemek için girişlerin davranışlarını modelleyen test modülleri hazırlanmıştır. Test modülü diğer alt modüllerden ve/veya sistem dışından gelen uyartıları içerir. Cadence DF II (Design Framework II) tasarım ortamının temel lojik simülatörü Verilog XL kullanılarak tasarımın doğruluğu test modülleri ile sınanmıştır. Alt modüller ayrı ayrı test edildiklerinde simülasyon sonuçlarında problem varsa tekrar kod aşamasına dönülüp akıştaki işlemler tekrarlanır. Simülasyon sonuçları istenen çıkışlar ile uyumlu ise alt modüller birleştirilerek üst modül halinde tekrar sentezlenir. Hazırlanan uygun test vektörlerine göre üst modül simülatör yardımıyla sınanır. Test sonucu problemli ise kod düzeyindeki tasarıma dönülerek akış tekrarlanır. Simülasyon sonuçları istenen çıkışlar ile uyumlu ise bu sefer daha önce ayrılan ana hiyerarşiler birleştirilir. Sistem bazında sentezleme, test modülü oluşturma ve simülasyon işlemleri tamamlanır. Simülasyon sonuçlarında problem olmadığı tespit edilirse tasarımın FPGA ortamında test edilebilmesi için giriş-çıkış (I/O) ve test pinlerinin listesi hazırlanır. Tasarım FPGA ortamına aktarılır. FPGA kütüphanesi ile ana blok tekrar sentezlenir ve serimi gerçekleştirilir. Simülatör yardımıyla daha önce oluşturulan üst düzey test modülü ile tasarım sınanır. Test sonucu problemli ise kod düzeyindeki tasarıma dönülerek akış tekrarlanır. Simülasyon sonuçlarında problem olmadığı tespit edilirse test kartı üzerinde Logic Tester (LV500) ve bir PC yardımıyla devre sınanır. Gerçek ortamdaki test aşamasında alt hiyerarşilerin davranışlarını gözlemlemek üzere tasarım uygun versiyonlara ayrılır. Test sonucu elde edilen sonuçlar, istenen çıkışlar ile uyumlu değilse test edilen versiyona ait kod düzeyi tasarım aşamasına dönülür. Tüm versiyonlar için test sonucunda problem olmadığı tespit edilirse, sentezlenmiş USB arayüz dijital bloğu gerektiğinde kapı dizisi ve ya standart cell yapısında semicustom olarak tasarlanabilir.

5.2 Verilog Tanımlama Dili

Verilog, 1984 yılında Philip Moorby tarafından, daha sonra Cadence firması tarafından satın alınan Gateway bünyesinde geliştirilen donanım tanımlama dilidir. Dijital sistemleri oluşturan eleman sayısındaki yoğunluk (VLSI) ve karmaşıklıktan dolayı tasarımcılar tanımlama dili kullanarak sistem tasarımı yoluna gitmişlerdir. Verilog ve VHDL olmak üzere endüstride tasarımcıların kullandığı 2 temel tanımlama dili mevcuttur. Programlama dilini andıran yapısı ile Verilog tanımlama dilinin kullanımı ve öğrenimi daha kolaydır. Günümüzde özellikle Amerika kıtasında en popüler donanım tanımlama dili olarak kabul edilmekte ve tüm dünyada yaygın bir şekilde kullanılmaktadır. Verilog dilinde, dijital bir sistemin uygun alt modüllere ayrıştırılarak tanımlanması yaklaşımı benimsenmiştir.

Hiyerarşide yer alan her modül iki farklı boyutta tanımlanır.

- Diğer modüller ile olan arayüzü
- Kendi lojik iç yapısı ve/veya davranışı

Dijital bir modülü, Verilog HDL ile farklı biçimlerde tanımlamak mümkündür.

Davranışsal Tanımlama (Davranışsal) Sistemden istenilen davranışı lojik yapıdaki karşılığını düşünerek programlama dillerini andıran tarzda tanımlama.

Yapısal Tanımlama (Structural) Dijital modülün iç lojik yapısını oluşturan elementer lojik kapıların ve/veya çeşitli lojik işlevleri yerine getiren alt modüllerin birbirleriyle olan bağlantıları dikkate alınarak tanımlama.

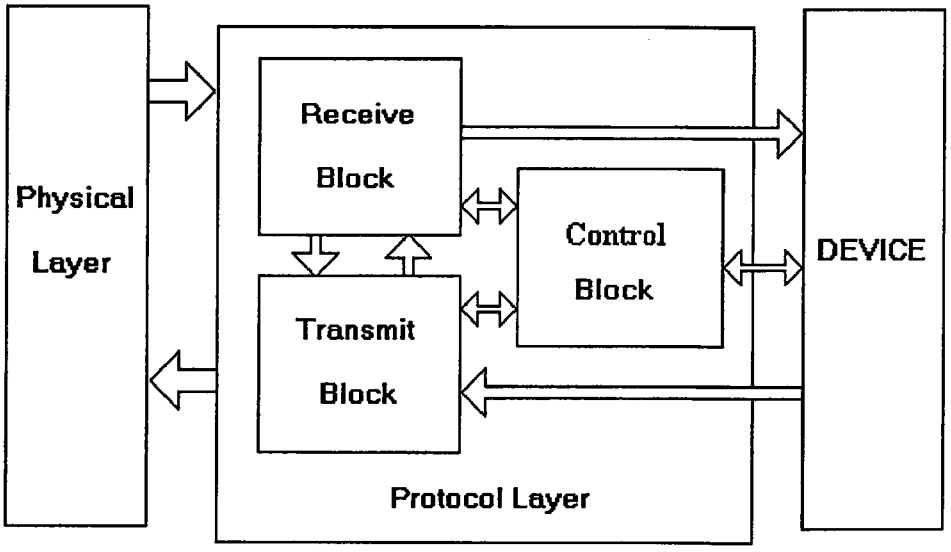
Çoklu Tanımlama (Mixed Mode) Aynı modül içerisinde her iki tanımlama biçimini (davranışsal + yapısal) kullanarak tanımlama.

Dijital bir modül, tanımlama biçiminden bağımsız olarak kombinezonsal ve / veya ardışıl lojik elemanlarından oluşur. Kombinezonsal lojik tanımlayan blok, girişlerinden herhangi birinin değişmesi durumunda gerçekleştirilecek işlemleri tanımlayan ifadeleri içermelidir. Ardışıl devreler 'Register Transfer Level' (RTL) yaklaşımı ile tanımlanırlar. Ardışıl bir devreye ilişkin RTL modeli, bir saat işaretinin yükselen ya da düşen kenarlarında sistem yazmaçları üzerinde gerçekleştirilmesi gereken işlemleri tanımlar. Bütünüyle kombinezonsal bir devrenin tersine, ardışıl bir tasarımda tanımlanan işlemler sürekli olarak (Saatin kenarı dışında) aktif değildirler.

6. PROTOKOL KATMANI

Bu bölümde İTÜ ETA ASIC tasarım merkezinde yürütülmekte olan USB cihaz arayüzü gerçekleştirme projesinin iki ana bölümden birini oluşturan protokol katmanı incelenecektir. Bu katmanda paket bazında değerlendirilen verinin işlenmesi, hata kontrolü, gerçek verinin ayıklanarak USB cihaza gönderilmesi, benzer şekilde cihazdan gelen gerçek verinin paketlenip elektriksel katmana gönderilme mekanizmaları anlatılacaktır. Cihaza veri aktarımında arabirim olarak kullanılan FIFO hakkında genel bilgi verilecektir. Ayrıca cihazın sisteme bağlanışı ve konfigürasyon aşamasında USB cihazın davranışı incelenecektir. Bölümün sonunda sentezleme adımı sonucu elde edilen simülasyon sonuçları sunulacaktır. İlk olarak 2. Bölümde bahsedilen veri paketi ve işlem birimi hakkında ayrıntılı bilgi verilecektir.

Protokol katmanı, fiziksel katman ile USB cihaz arasındaki veri iletiminde ara yüzey oluşturur (Şekil 6.1). Cihaza veri göndermekle ve cihazdan aldığı veriyi paket haline getirip fiziksel katmana iletmekle sorumludur. Hata kontrolü mekanizması ile alınan verinin nasıl değerlendirildiği hakkında gönderici tarafa bilgi vermek amacı ile farklı tiplerde veri paketleri (handshake) tanımlar. Verinin içeriği ile ilgilenen protokol katmanı, alınan verinin tipine bakarak hangi cihaz ve uç noktanın (endpoint) veri iletiminde kaynak veya hedef olarak rol alacağını belirler. Uç nokta, host ile cihaz arasındaki haberleşme akışının USB cihaz tarafındaki tanımlanabilir son noktasıdır.

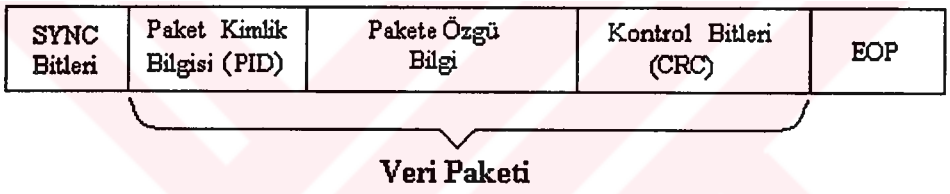


Şekil 6.1 USB cihaz ara yüzünde protokol katmanının konumu

Şekil 6.1'de, USB cihaz ara yüzeyine ait protokol katmanının en genel anlamda blok diyagramı verilmiştir. Cihaz ara yüzeyinde bulunan protokol katmanı, alıcı bloğu (receive), gönderici bloğu (transmit) bloğu ve kontrol bloğu olmak üzere üç ana grupta incelenir. Veri paketi ve işlem birimleri kavramları bu bölümde sıkça kullanılacağından 2. bölümde bahsedilen bu kavramları daha ayrıntılı biçimde açıklamakta fayda vardır.

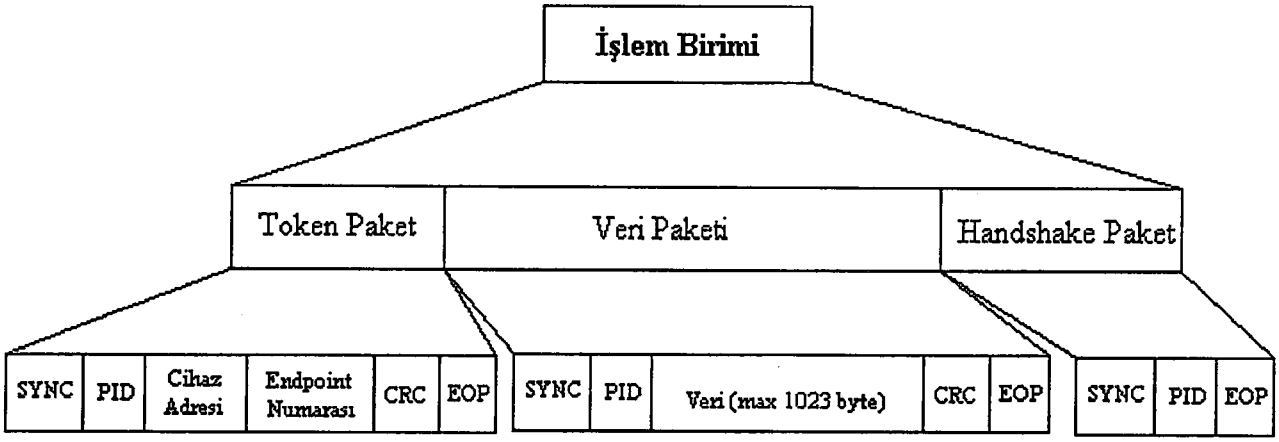
6.1 Veri Paketlerinin Yapısı

2. bölümde veri paketlerinin host denetleyici tarafından üretilmesi ve USB hatta kök hub tarafından çıkarılması hakkında bilgi verilmişti. Protokol katmanında veri, paket bazında değerlendirildiğinden veri paket yapısının anlaşılması çok önemlidir. En genel halde tüm veri paketleri aynı yapıdadır. Paketin başını ve sonunu gösteren, kimliğini belirleyen, pakete özgü bilgiyi ileten, alıcı tarafta hata kontrolünü sağlayan özel verilere sahiptir (Şekil 6.2).



Şekil 6.2 En genel ifadeyle veri paket yapısı

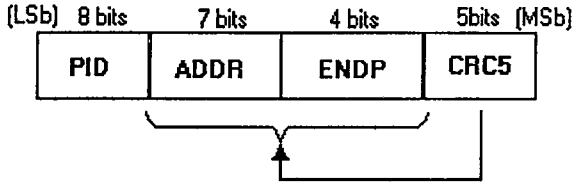
Veri paketlerinin bir araya gelmesi ile belirli amaca hizmet eden işlem birimleri oluşur. İşlem birimleri transfer tipine, hatalı veri iletimine bağlı olarak bir, iki ve ya üç veri paketini içerirler. Tipik olarak bir işlem birimi (Transaction) token paket, veri paketi ve el sıkışma (handshake) paketi olmak üzere 3 fazdan oluşur (Şekil 6.3).



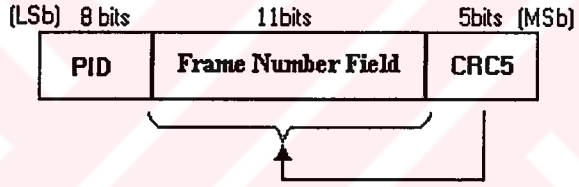
Şekil 6.3 3 fazdan oluşan işlem birimi yapısı

6.1.1 Token Paket Fazı

Her işlem birimi, işlem biriminin tipini belirleyen token paket fazı ile iletime başlar. IN, OUT, SETUP, SOF paketi olmak üzere 4 farklı token paket tipi mevcuttur. Paket tipi, paketin başında gelen ve paketin kimliğini belirleyen (PID) veri yapısı ile belirlenir. PID ve CRC bitleri dışında kalan veri alanı IN, OUT, SETUP veri paketleri için cihaz adresi ve uç nokta numarasını(Şekil 6.4), SOF veri paketi için çerçeve (frame) numarasını (Şekil 6.5) belirler.



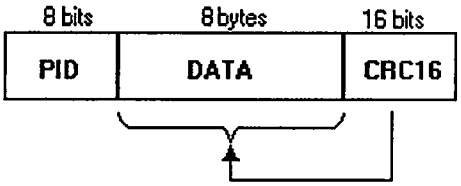
Şekil 6.4 IN, OUT, SETUP PID veri tipleri için paket yapısı (Compaq,1998)



Şekil 6.5 SOF PID veri tipi için paket yapısı (Compaq,1998)

6.1.2 Veri Paketi Fazı

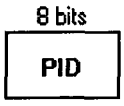
Token paketi ile seçilen cihaza veri iletmekle ve konfigürasyon aşamasında host ile USB cihaz arayüzü arasındaki iletimden sorumludur. Veri paketinin bir seferde taşıyabileceği maksimum veri uzunluğu transfer tipine bağlı olarak konfigürasyon aşamasında belirlenir. Fakat tasarımda 8 byte'lık FIFO kullanıldığından tüm transfer tiplerinde maksimum veri iletim boyutu 8 byte ile sınırlandırılmıştır. Senkronizasyon mekanizmasını destekleyen transfer tipleri için farklı PID bilgisine sahip DATA0 ve DATA1 veri paketleri vardır. Veri paket yapısı Şekil 6.6'te gösterilmiştir.



Şekil 6.6 Veri paket yapısı (Compaq,1998)

6.1.3 El Sıkışma (Handshake) Paket Fazı

El Sıkışma paketleri USB cihaz arabiriminin durumu hakkında host'a ve iletilen verinin alıcı tarafta hatasız alındığına dair gönderen tarafa bilgi vermek amacıyla kullanılır. Farklı amaçlar için kullanılan ve birbirlerine göre öncelikleri olan ACK, NAK, STALL olmak üzere 3 farklı el sıkışma paket tipi mevcuttur. El sıkışma veri paketleri 1byte'lık PID verisinden oluşur. El sıkışma paket yapısı Şekil 6.7'da gösterilmiştir.



Şekil 6.7 El sıkışma (Handshake) veri paket yapısı (Compaq,1998)

SYNC ve EOP, sırasıyla bir paketin başladığını ve bittiğini gösteren, paketin başında ve sonunda bulunan verilerdir. USB hattında SYNC bilgisinin algılanması yeni bir paketin başladığı anlamına gelir. SYNC bilgisi gelen veri ile alıcının kullandığı saat işaretinin senkronizasyonunu sağlar. USB hattında EOP (End of Packet) verisinin elde edilmesi ise bir paketin sonlandırıldığı anlamına gelir. Fiziksel katmanda senkronizasyon ve SYNC, EOP algılama işlemleri tamamlandığından, protokol katmanına bu veriler gönderilmez.

Paket tipleri, veri yapıları hakkında verilen ön bilgiler doğrultusunda tasarımda kullanılan alt blokların üstlendikleri görevler daha iyi anlaşılacaktır.

6.2 Alıcı (Receive) Ana Bloğu

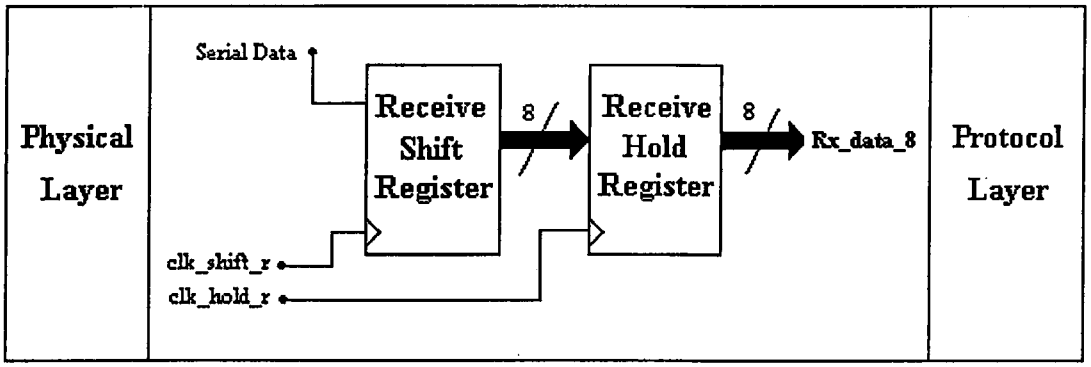
Alıcı ana bloğu, host tarafından iletilen veri paketini işleyip, gerektiğinde cihaza göndermekle görevlidir. USB cihaz arayüzeyinin fiziksel katmanında ön işlemlerden geçen veri, protokol katmanının alıcı ana bloğunda alınan veri paketinin tipine bağlı olarak farklı işlem bloklarından geçirilir. Takip eden bölümde, alıcı ana bloğunda bulunan veri işlemekle görevli bloklar incelenecektir. Fiziksel katman ile arabirim oluşturan Şekil 5.1'de gösterilen arayüz yazmaçlarından gelen kontrol işaretlerin davranışları dikkate alınarak alıcı alt blokları tasarlanmıştır. Arayüz yazmaçlarından gelen kontrol işaretleri vasıtasıyla fiziksel katman ile protokol katmanı arasındaki senkronizasyon ve kontrol sağlanabilir.

6.2.1 Alıcı Bloğuna gönderilen Arayüz Kontrol ve Saat İşaretleri ile İletim Hattı

- **Rx_Valid :** Fiziksel katmanda işlenen veriyi tutmakla görevli arayüz yazmaçlarının durumu hakkında protokol katmanına bilgi verir. İşlenen veri arayüz yazmaçlarına yüklenir yüklenmez Rx_Valid bayrağı 1'e çekilir. Protokol katmanı yazmaçlarda bulunan veriyi çekmek için clk_hold_r saat işaretinin 1'e çekilmesini bekler. Protokol katmanı tarafından

son veri çekildiği anda Rx_Valid bayrağı 0'a çekilir. Böylece protokol katmanı işlenecek son veriyi aldığını bilir.

- **Rx_data_8 :** Fiziksel katmandan, protokol katmanına veri iletimi sırasında arayüzeyde öteleme yazmacı (shift register) ve tutma yazmacı (hold register) kullanılmaktadır (Şekil 6.8). Bu yazmaçlar USB hattından gelen fiziksel katmanda işlenen seri veriyi, paralel veriye çevirip protokol katmanına göndermekle görevlidir. Her iki yazmaç 8 bitliktir. Öteleme yazmacı seri veriyi paralele çevirmek için, tutma yazmacı alınan veriyi tutmak ve protokol katmanına aktarmak için kullanılır. Tutma yazmacı ile protokol katmanı arasındaki veri iletiminde Rx_data_8 hattı kullanılır.



Şekil 6.8 Protokol katmanına veri iletmekle görevli arayüz yazmaçları

- **clk_hold_r :** Öteleme yazmacı (Shift Register) çıkışında elde edilen paralel verinin Rx_data_8 hattına yüklenmesinden sorumludur. Belirli bir durumda 'clk' saat işaretinin düşen kenarıyla birlikte 'clk_hold_r' saat işareti 1'e , 'clk' işaretinin diğer düşen kenarında ise clk_hold_r tekrar 0'a çekilir. Protokol katmanı, clk_hold_r saat işareti 1' çekildikten sonra gördüğü 'clk' sinyalinin ilk yükselen kenarında tutma yazmacına yüklenen paralel veriyi çeker.

- **clk :** Arayüz yazmaçlarında elde edilen paralel veriyi 'clk_hold_r' saat işaretinin kontrolü altında protokol katmanına yüklemekten sorumludur. Bu saat işareti fiziksel katmanda bulunan DPLL bloğunun çıkışından elde edilir.

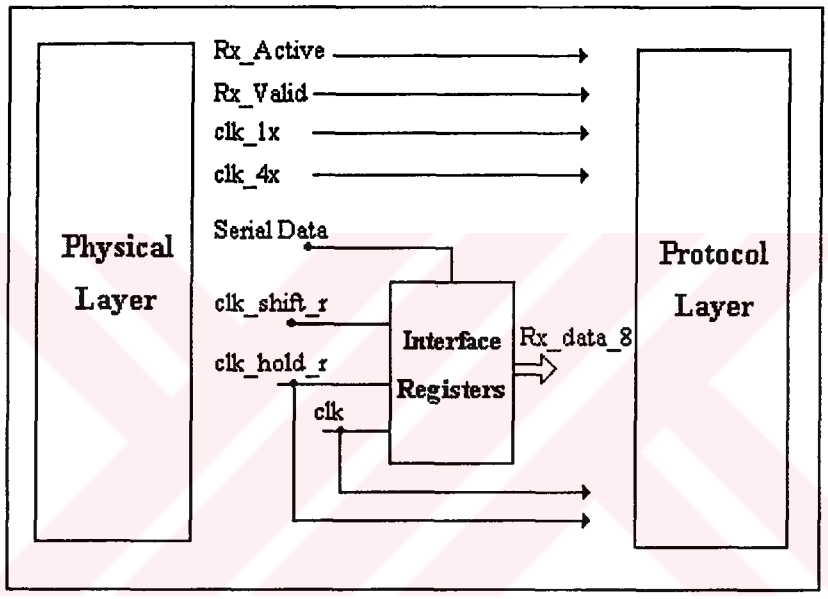
- **Rx_Active :** Protokol katmanına USB hattından veri gelip gelmediğini haber verir. USB hattı half duplex modda çalıştığından aynı anda karşılıklı veri alış verişi söz konusu değildir. Bu nedenle cihaz tarafından USB hattına veri çıkarılmadan (Transmit) önce bu sinyal kontrol edilir.

- **clk_1x & clk_4x :** Fiziksel katmanda bulunan 'saat üreteci' (Clock Generator) bloğunun çıkışında elde edilen sabit frekanslı saat işaretleridir. clk_4x saat işaretinin frekansı, clk_1x işaretinin frekansının 4 katı olarak üretilmektedir. Çalışma hızına (Low speed & Full Speed) bağlı olarak çalışma frekansları değişmektedir. Bu işaretler alıcı ana bloğunun alt bloklarında kullanılmaktadır. Çizelge 6.1'de saat işaretlerinin çalışma hızına bağlı frekansları verilmiştir. (USB Tasarım Grubu, 2001).

Çizelge 6.1 Saat işaretlerinin çalışma hızına bağlı frekansları

	clk_1x	Clk_4x
Düşük Hızlı	1.5MHz	6MHz
Orta Hızlı	12MHz	48MHz

Veri alma işlemi sırasında arayüz ile kontrolü sağlayan işaretler Şekil 6.9'de gösterilmiştir.



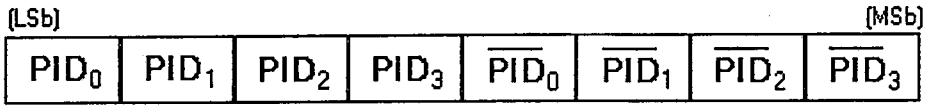
Şekil 6.9 Veri alma işlemi sırasında arayüz ile kontrolü sağlayan işaretler

6.2.2 PID Kontrol ve PID Kod Çözücü Bloğu

PID verisi, protokol katmanına iletilen her verinin başında bulunur. Gelen 8 bitlik PID verisinin hatasız iletildiği PID kontrol bloğunda tespit edildiğinde 8 bitlik paralel veri, PID tipini belirlemek üzere PID kod çözücü bloğuna geçirilir.

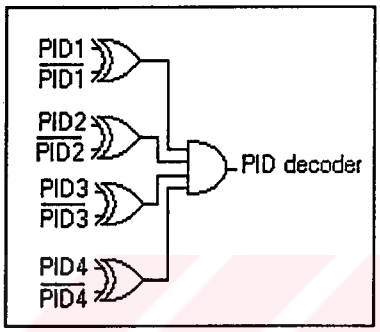
6.2.2.1 PID Kontrol Bloğu

Arayüz yazmaçlarından elde edilen 8 bitlik paralel veri, kombinezonsal bloğunun girişlerine gelir. Yazmaçlardan alınan veri, PID verisi ise, 8 bitlik verinin en az ağırlıklı ilk 4 biti paketin tipini (kimliğini) belirler. Diğer 4 bitlik veri alanı ise PID verisinin doğruluğunu kontrol eder. Kontrolü sağlayan 4 bitlik veri, paketin tipini belirleyen 4 bitlik veri alanının tümleyenidir. 8 bitlik PID verisi Şekil 6.10'da gösterilmiştir.



Şekil 6.10 PID veri alanı ve kontrol bitleri (Compaq,1998)

Kontrol işlemi Şekil 6.11'da gösterilen kombinezonsal blok ile sağlanır. Kontrol sonucunda PID verisi bozulmuş ise kombinezonsal bloğun çıkışı '0' değerini alacaktır. Böylece PID kod çözücü bloğu aktif hale getirilemeyecektir. PID hatasız iletildi ise, kontrol bloğunun çıkışı 1'e çekilecektir.



Şekil 6.11 PID kontrol işlemini gerçekleyen kombinezonsal blok

6.2.2.2 PID Kod Çözücü Bloğu

Arayüz kontrol işareti Rx_Valid 'in 1'e çekilmesi ile protokol katmanı clk_hold_r saat işaretinin lojik 1 seviyesine çıkmasını bekler. clk_hold_r sinyalinin 1'e çıkması ile 'clk' işaretinin ilk yükselen kenarında protokol katmanı tarafından çekilen veri PID'dir. Bu anda PID kontrol bloğu çıkışında elde edilen işaret lojik '1' değerinde ise PID kod çözücü (4 × 16) bloğu aktif hale gelir. Gelen paketin tipine bağlı olarak kod çözücü çıkışlarından biri 1'e çekilir. Kod çözücü bloğu, 8 bitlik PID verisinin en az ağırlıklı ilk dört bitine göre paket tipini belirler. 16 çıkışın sadece 8 tanesine protokol katmanı tarafından paket tipi atanmıştır. Diğer 8 tanesi default değeri olarak kullanılır. Default değerlerinden biri kod çözücü çıkışında aktif ise gelen veri işlenmez. PID paket tipleri token, veri, el sıkışma ve özel (Special) olmak üzere 4 ana grupta incelenir. Çizelge 6.2'de kod çözücü çıkışında geçerli olan PID tipleri, PID kodları ve paketi gönderen uç birimler belirtilmiştir.

PID kod çözücü çıkışında token paket tiplerinden;

- PID_IN çıkışı aktif hale gelirse, paket yapısında yer alan adres alanı ile belirlenen cihaz ve cihaza ait uç nokta bir sonraki veri paketini host'a gönderir.

- PID_OUT çıkışı aktif hale gelirse, paket yapısında yer alan adres alanı ile belirlenen cihaz ve cihaza ait endpoint bir sonraki veri paketini alır.

- PID_SETUP çıkışı aktif hale gelirse, ardından gelecek veri paketi ile konfigürasyon işlemleri başlatılır. Diğer paket tiplerine göre önceliklidir.

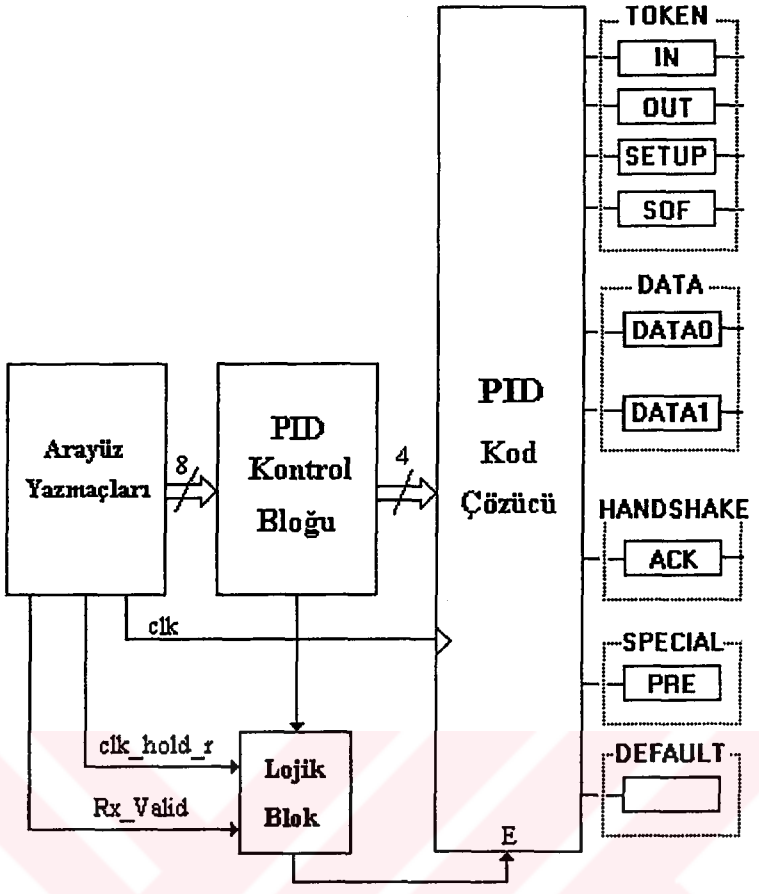
- PID_SOF çıkışı aktif hale gelirse, PID'nin ardından gönderilen çerçeve numarasını belirten 11 bitlik senkronizasyon bilgisi orta hızlı cihazlar tarafından alınır.

Çizelge 6.2 PID tipleri, PID kodları ve paketi gönderen uç birimler

PID Tipi	PID İsmi	PID[3:0]	Gönderen HOST	Gönderen USB CİHAZ
TOKEN	OUT	0001B	√	×
	IN	1001B	√	×
	SOF	0101B	√	×
	SETUP	1101B	√	×
DATA	DATA0	0011B	√	√
	DATA1	1011B	√	√
HANDSHAKE	ACK	0010B	√	√
	NAK	1010B	×	√
	STALL	1110B	×	√
SPECIAL	PRE	1100B	√	×

DATA0 ve DATA1 paket tipleri ile gerçek veri iletimi sağlanır. Kod çözücü çıkışında el sıkışma paket tiplerinden PID_ACK aktif ise host'a gönderilen verinin başarılı bir şekilde sonuçlandırıldığı USB cihaza bildirilir. Kod çözücü çıkışında STALL ve NAK PID tipleri için var olan çıkışlar hiçbir zaman aktif hale getirilemez. Çünkü bu tür el sıkışma paketleri host tarafından gönderilemez.

Kod çözücü çıkışında özel (Special) paket tipi olarak sayılan PRE PID paket tipi aktif ise USB hub, bu paket ardından gönderilecek veri paketinin düşük hızlı olduğunu anlar ve düşük hızlı portlarını aktif hale getirir. PRE PID paket tipi USB cihazlar tarafından tanımlı değildir.



Şekil 6.12 PID Kontrol ve kod çözücü Bloğu

6.2.3 CRCs (=Cyclic Redundancy Checks) İşlem Bloğu

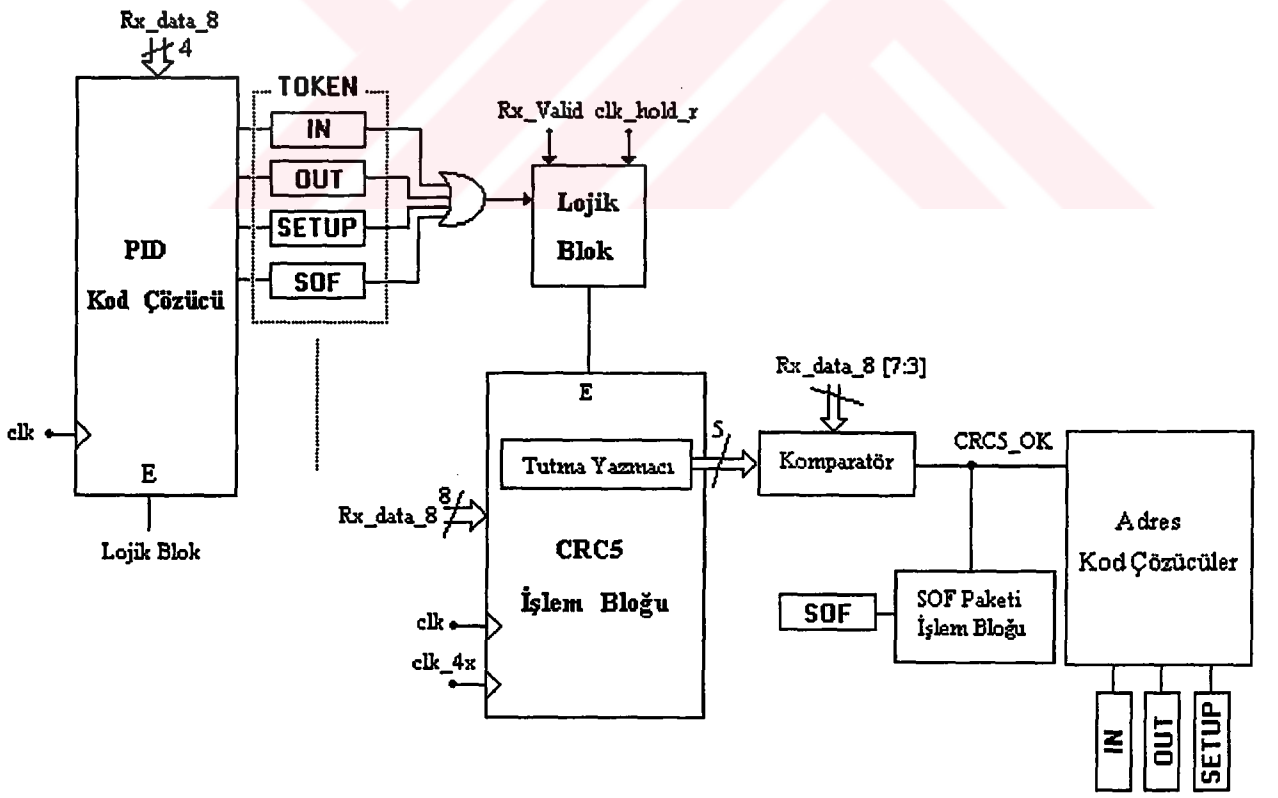
CRC işlemi token ve veri paketi içindeki, PID alanı dışında kalan verilerin doğruluğunu kontrol eder. Gelen veri veya token paketi işlenmeden önce CRC kontrol bloğundan geçirilir. Hatalı sonuç gelen paketin işlenmemesine neden olur. Benzer şekilde host'a veri göndermeden önce CRC bitleri üretmek üzere aynı işlem bloğundan geçirilir. Elde edilen CRC bitleri gerçek verinin ardından USB hatta çıkarılır. Alıcı tarafta CRC hatası tespit edildiğinde, bu durum gönderici tarafa bildirilmez. Veri gönderen taraf zaman aşımı (timeout) mekanizması kullanarak alıcı tarafta hata tespit edildiğini algılar.

6.2.3.1 CRC5 Kontrol Bloğu

PID kod çözücü çıkışında, token paket tiplerinden herhangi biri seçildiğinde CRC5 kontrol bloğunun aktif hale gelebilmesi için clk_hold_r saat işaretinin lojik '1' seviyesine çekilmesi beklenir. clk_hold_r sinyalinin 1'e çıkması ile 'clk' işaretinin ilk yükselen kenarında protokol katmanı tarafından çekilen veriye CRC5 kontrol işlemi uygulanmaya başlanır. CRC5 kontrol işlemi 11 bitlik veri üzerinde uygulanır. CRC5 işlemi clk_4x saat işaretini kullanarak

gerçekleşir. CRC5 kontrol işlemi adımları aşağıda sıralı olarak belirtilmiştir.

- 5 bitlik CRC5 tutma (Hold) yazmacı lojik '1' ile yüklenir.
- Veri ile tutma yazmacının en fazla ağırlıklı (MSB) bitleri EXOR işleminden geçirilir.
- Yazmaç 1 bit sola kaydırılır (shift) ve en az ağırlıklı bitine lojik '0' değeri yüklenir.
- EXOR işlemi sonucu lojik '1' ise bu sefer tutma yazmacı ile üreteç (generator) polinom EXOR işleminden geçirilir. EXOR işlemi sonucu lojik '0' ise yazmaç değerini korur.
- Token paket için; generator polinom = 10101B
- Token paketin PID ve CRC verileri dışında kalan 11 bitlik veriye yukarıdaki işlemler sırasıyla uygulanır.
- Verinin tüm bitleri uygulandığında CRC5 kontrol işlemi sona erer ve tutma yazmacında en son elde edilen verinin tümleyeni alınır.
- Tümleyeni alınan tutma yazmacındaki veri ile veri paketi içinde elde edilen CRC bitleri karşılaştırılır. Karşılaştırma sonucu veriler birbirine eşit ise USB hat üzerinden iletilen token veri paketi hatasız alınmıştır.
- CRC5 işlemi hatasız sonuçlandırıldığında token paket tipi IN, OUT ve ya SETUP PID tiplerinden biri ise; hedef cihaz ve uç noktayı belirlemek üzere adres kod çözücü bloğu aktif hale getirilir.
- CRC5 işlemi hatasız sonuçlandırıldığında token paket tipi SOF PID ise; host'tan gönderilen çerçeve (frame) numarasını işlemek üzere SOF (Start of Frame) işlem bloğu aktif hale getirilir.



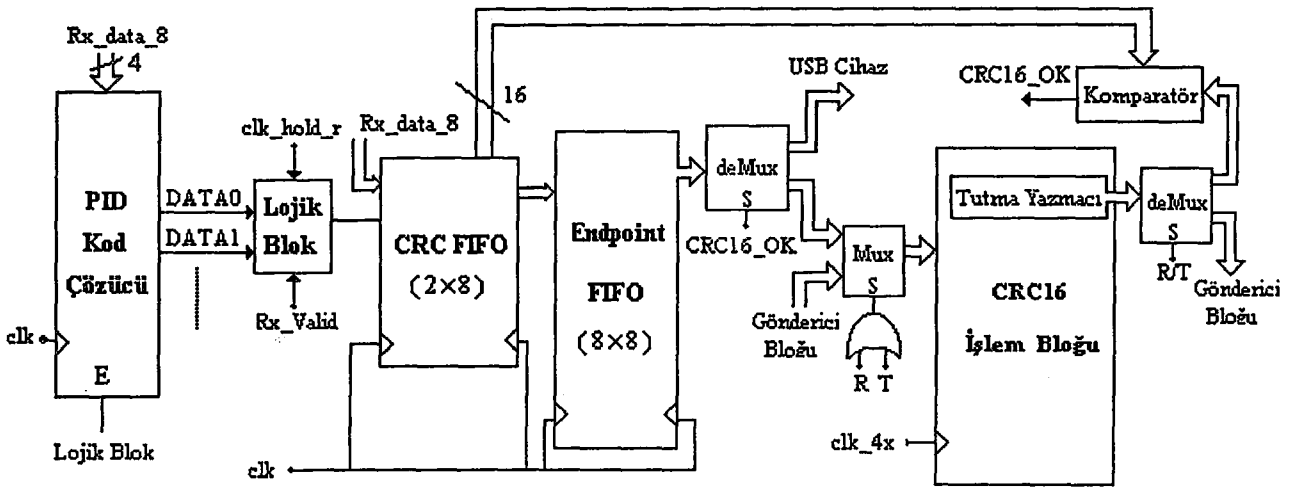
Şekil 6.13 CRC5 işlem bloğu

6.2.3.2 CRC16 Kontrol Bloğu

DATA0 ve ya DATA1 PID tiplerinden biri PID kod çözücü bloğunun çıkışında aktif ise gönderilen veri paketi arayüz yazmaçlarından clk_hold_r ve clk saat işaretleri kullanılarak protokol katmanı tarafından çekilir. Hattan alınan veri paketinin uzunluğu bilinmediğinden veriden CRC bitlerini ayırmak üzere (2×8) FIFO yazmacı kullanılır. Rx_Valid bayrağı lojik '1' değerinde olduğu sürece arayüz yazmaçlarından çekilen veri, ilk önce CRC bitlerini ayıran (2×8) FIFO yazmacına, ardından ilgili uç noktanın (8×8) FIFO yazmacına yazılır. Arayüz yazmaçlarından çekilen verinin ilk byte'ının Endpoint (8×8) FIFO yazmacına alınması ile birlikte CRC16 kontrol bloğu aktif hale getirilir. Böylece kontrol işlemi hattan alınan veriye senkron, gecikmesiz olarak gerçekleşir. Arayüz yazmaçlarından çekilen veri paketinin son byte'ı CRC bitlerini ayıran (2×8) FIFO yazmacına yüklendiğinde Rx_Valid kontrol işareti lojik '0' değerine çekilir.

CRC16 kontrol işlemi arayüz yazmaçlarından alınan n × 1 byte uzunluklu verilere uygulanır. n sayısı tamsayı olup FIFO boyutuna bağlı olarak maksimum 8 değerini alır. CRC16 işlemi de clk_4x saat işaretini kullanarak gerçekleşir. CRC16 kontrol işlemi adımları aşağıda sıralı olarak belirtilmiştir.

- 16 bitlik CRC16 tutma (Hold) yazmacı lojik '1' ile yüklenir.
- Veri ile tutma yazmacının en fazla ağırlıklı (MSB) bitleri EXOR işleminden geçirilir.
- Yazmaç 1 bit sola kaydırılır (shift) ve en az ağırlıklı bitine lojik '0' değeri yüklenir.
- EXOR işlemi sonucu lojik '1' ise bu sefer tutma yazmacı ile üreteç (generator) polinom EXOR işleminden geçirilir. EXOR işlemi sonucu lojik '0' ise yazmaç değerini korur.
- Veri paketi için; generator polinom =1000000000000101B
- Veri paketinin PID ve CRC verileri dışında kalan n × 1 byte uzunluklu veriye yukarıdaki işlemler sırasıyla uygulanır.
- Verinin tüm bitleri uygulandığında CRC16 kontrol işlemi sona erer ve tutma yazmacında en son elde edilen verinin tümleyeni alınır.
- Tümleyeni alınan tutma yazmacındaki veri ile veri paketi içinde elde edilen CRC bitleri karşılaştırılır. Karşılaştırma sonucu veriler birbirine eşit ise USB hat üzerinden iletilen veri paketi hatasız alınmıştır.
- CRC16 işlemi hatasız sonuçlandırıldığında, cihazın durumu hakkında ve ya veri iletiminin sonucu hakkında host'a bilgi vermek amacıyla handshake bloğu aktif hale getirilir. Eğer CRC16 kontrol işlemi sonucu hata tespit edilmişse host'a bilgi verilmez. Host zaman aşımı (Timeout) mekanizmasını kullanarak gönderdiği verinin cihaza hatalı ulaştığını algılar ve aynı veriyi aynı çerçevede veya başka çerçevede tekrar gönderir.

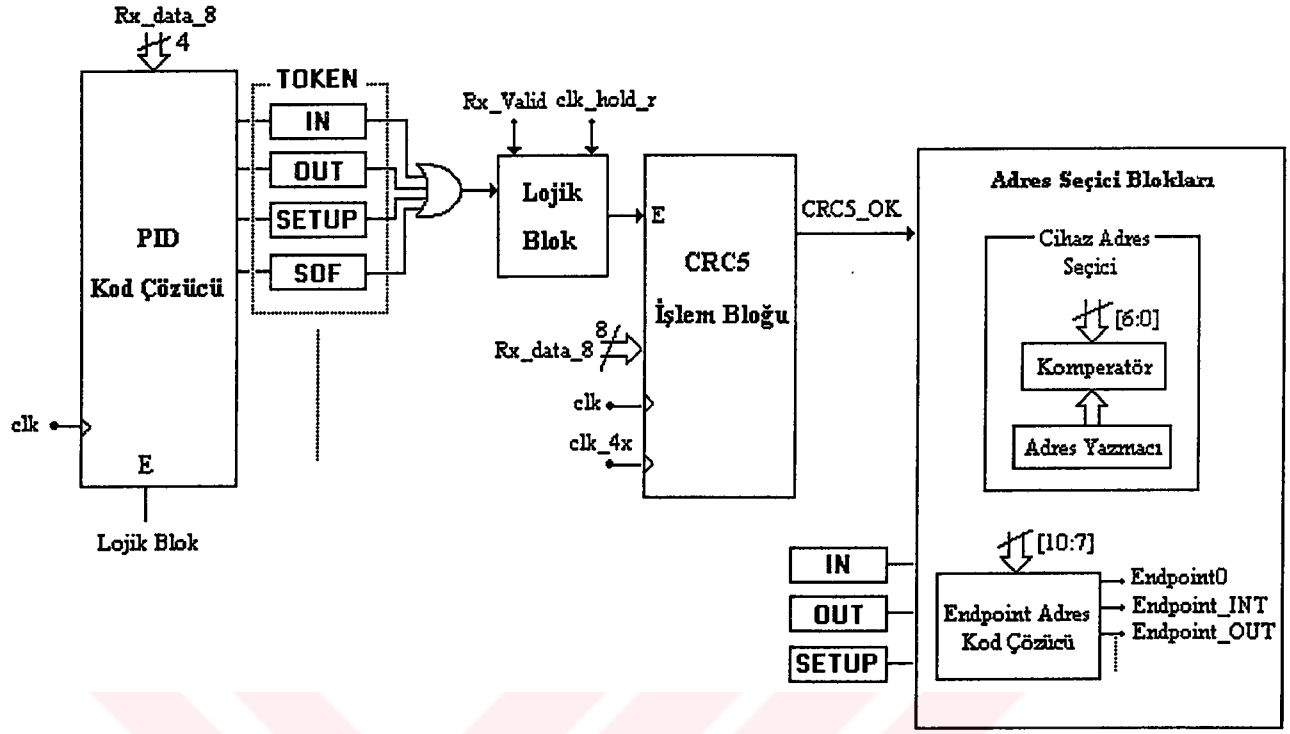


Şekil 6.14 CRC16 işlem bloğu

6.2.4 Adres Kod Çözücü Bloğu

CRC5 kontrol işlemi sonucu token veri paketinin (PID_IN, PID_OUT, PID_SETUP) alıcı tarafa hatasız olarak iletildiği belirlendiğinde adres kod çözücü bloğu aktif edilir. Şekil 6.4'te gösterilen token paket yapısının (IN, OUT, SETUP) 11 bitlik adres alanının en az ağırlıklı ilk 7 biti seçilecek cihaz adresini, son 4 biti ise seçilecek cihaza ait uç nokta numarasını gösterir. Cihaz adres bilgisi ile uç nokta adres bilgisi, adres kod çözücü bloklarından geçirilir. Uç nokta adres kod çözücü ve cihaz adres kod çözücü çıkışında sisteme bağlı cihazlardan biri ve seçilen cihaza ait uç nokta seçilir. Cihaz adres kod çözücü çıkışında konfigürasyon sırasında adreslenmiş 127 (2^7-1) cihazdan herhangi biri hattan gelen adres bilgisi doğrultusunda seçilir. (7'b0000000 adresi konfigürasyon sırasında kullanılır.)

Endpoint adres kod çözücü bloğunun çıkışında düşük hızlı cihazlar için 3 uç nokta adresinden herhangi biri, orta hızlı cihazlar için 16 uç nokta adresinden herhangi biri seçilir. Tasarımda düşük hızlı cihazları destekleyen USB cihaz arayüzü tasarlandığından 3 farklı uç nokta adresi kullanılmaktadır.



Şekil 6.15 Adres kod çözücü bloğu

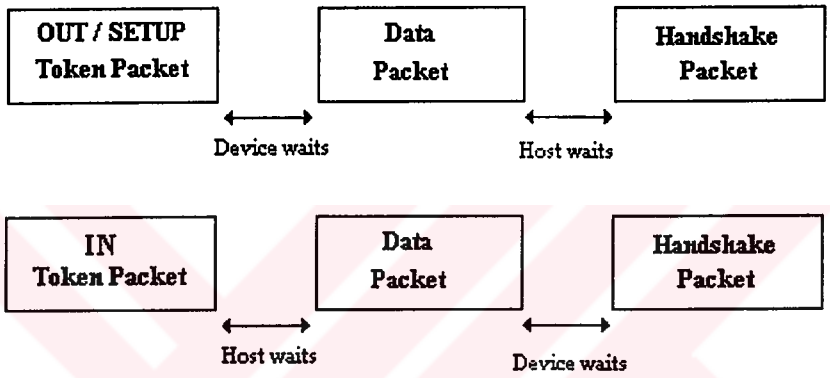
6.2.5 Timeout Kontrol Bloğu

Hattan alınan veri paketi OUT ve ya SETUP PID tiplerinden biri ise CRC5 kontrol bloğundan geçirilerek verinin hata durumu kontrol edilir. Verinin hatasız alındığı tespit edildiğinde hedef cihaz ve uç noktayı belirlemek üzere veri paketi adres kod çözücülerden geçirilir. İşlem birimi (Transaction) yapısına bağlı olarak OUT ve SETUP token paketlerinin ardından host'tan gönderilmek üzere veri paketi (DATA0/DATA1) beklenir. Belirli bir süre içinde hattan, beklenen veri paketi gelirse, alınan veri, token paketi ile seçilen uç noktanın FIFO yazmacına yazılır. Eğer beklenen veri belirlenen süre sonunda gelirse ve ya hiç gelmez ise alıcı bloğu tarafından zaman aşımına uğrar. Host'a veri alındığını bildirmek üzere veri paketi (handshake) iletilmez. Bu durum için belirlenen süre 'Zaman Aşımı (Timeout) Süresi' olarak adlandırılır. Zaman aşımı süresi burada OUT ve SETUP token paketleri ardından gönderilecek veri paketi için cihaz tarafındaki bekleme zamanını sınırlar.

Gönderici bloğunda işlem birimi yapısına bağlı olarak IN paketinin ardından host'a veri paketi gönderilir. Host, aldığı veri paketini kontrol işlemleri sonucu hatasız değerlendirdiğinde cihaza bildirmek üzere el sıkışma paketi yollayacaktır. USB cihaz arayüzü, host tarafından gönderilecek olan el sıkışma paketini belirli süre içinde hattan almayı bekler.

Zaman aşımı süresi burada host tarafından gönderilecek el sıkışma paketi için cihaz tarafındaki bekleme zamanını sınırlar.

Zaman aşımı süresi, hub'lardaki ve kablolardaki yayılım gecikmeleri ile hedef cihazın cevap verme süresindeki gecikme dikkate alınarak hesaplanmıştır. Bu süre 16 bit süresinden daha kısa, 18 bit süresinden daha uzun olamaz. Zaman aşımı süresi host ile cihaz tarafından tanımlıdır. Host, cihaz tarafından gönderilecek veri paketlerini belirli zaman içinde almayı ister. Cihaz ise aynı şekilde, host tarafından gönderilecek veri paketlerini belirli zaman içinde almayı ister. Bu durum Şekil 6.16'de açıkça gösterilmiştir.



Şekil 6.16 Zaman aşımı süresi içinde cihaz veya host tarafından beklenen veri paketleri (Compaq, 1998)

Zaman aşımı mekanizması, fiziksel katmandan gelen, hatta çıkarılan veya hattan alınan verinin başını ve sonunu belirleyen bayraklar doğrultusunda protokol katmanı tarafından tasarlanmıştır.

OUT veya SETUP işlem biriminde cihaz tarafından USB hattan alınan token paketin son verisi alındığında yani EOP'dan hattın idle durumuna geçiş (SE0_to_J_flag) sırasında timeout sayıcısı aktif hale gelir. Timeout sayıcısı clk_4x frekansı ile çalışmaktadır. Zaman aşımı süresi 16 ile 18 bit süresi arasında sınırlandırıldığından, sayıcı 70'e kadar sayma süresi içinde host'tan gönderilecek veri paketini bekler. Fiziksel katman tarafından veri paketine ait ilk veri alındığında, protokol katmanına bir kontrol bayrağı (SOP) ile bildirilir. Bu durum zaman aşımı süresi içinde gerçekleşirse, gelen veri paketi işlenmek üzere ilgili uç noktanın FIFO yazmacına aktarılır.

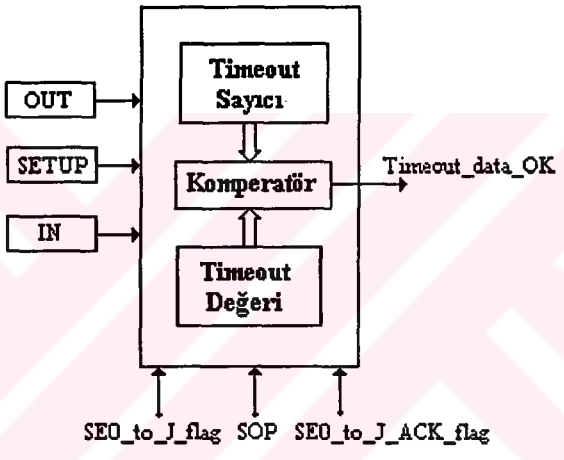
Benzer işlem IN işlem biriminde gerçekleşir (Şekil 6.12). IN token paketi ile belirlenen uç noktanın FIFO yamacındaki veri, paket haline getirilip fiziksel katman tarafından hatta çıkarıldığında, son verinin yüklenmesi ile birlikte timeout sayıcı bloğu aktif hale getirilir.

Zaman aşımı süresi içinde ($70 \times T_{clk\_4x}$), host'tan veri paketini hatasız aldığına dair el sıkışma paketi beklenir. Fiziksel katman tarafından el sıkışma paketine ait ilk veri alındığında, protokol katmanına kontrol bayrağı (SOP) ile bildirilir. Bu durum zaman aşımı süresi içinde gerçekleşirse, ilgili uç noktaya ait senkronizasyon bitinin tümleyeni alınır. Senkronizasyon bitinin neden kullanıldığına dair ayrıntılı bilgi takip eden bölümde işlenecektir. Timeout bloğunun tasarımında kullanılan kontrol bayrakları aşağıda sıralanmıştır. Bu kontrol bayrakları tasarımda clk_4x ile örneklenir.

SE0_to_J_flag : Hattan alınan veri paketinin son bitini algılar.

SOP (Start of Packet) : Hattan alınan veri veya el sıkışma paketinin ilk bitini algılar.

SE0_to_J_ACK_flag : Hatta çıkarılan verinin son bitini algılar.



Şekil 6.17 Timeout işlem bloğu

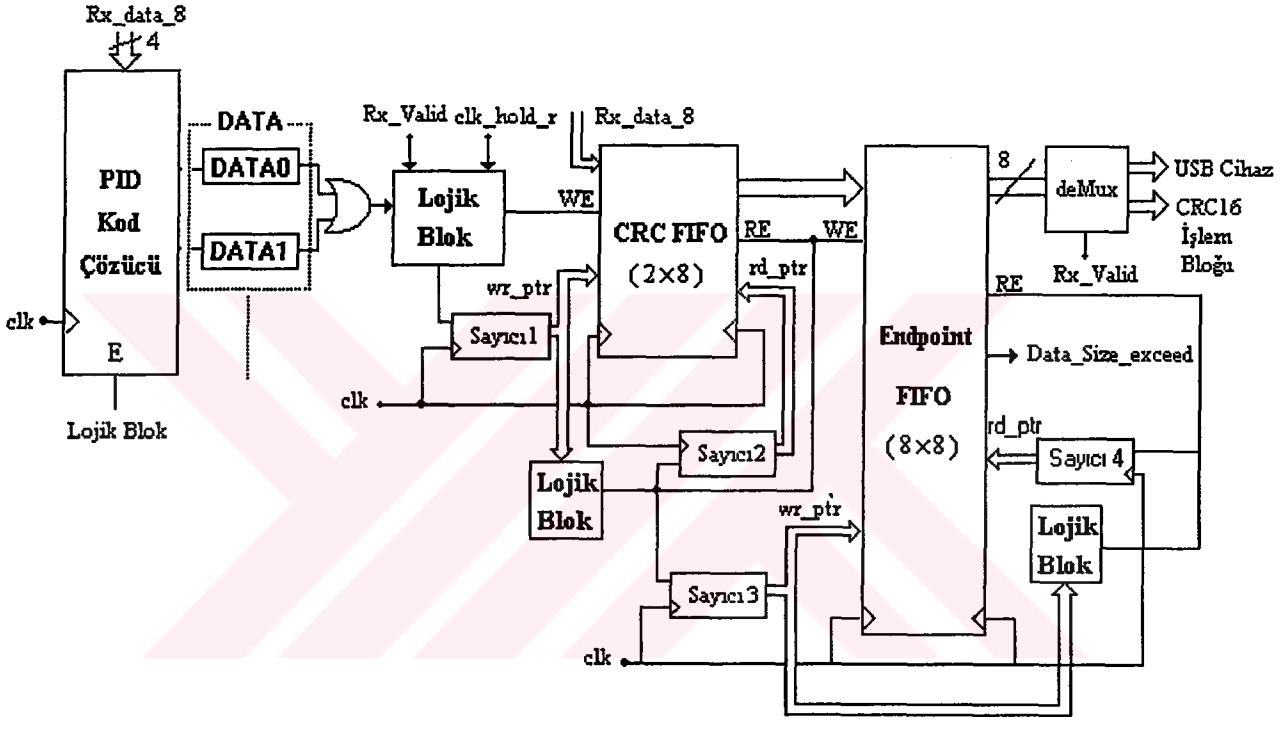
6.2.6 Alıcı Bloğunda Kullanılan Endpoint FIFO Yazmacı

Zaman aşımı süresi içinde gelen veri paketi işlenmek üzere OUT veya SETUP token paketi ile seçilen uç noktanın FIFO yazmacına aktarılır. Alınan verinin endpoint FIFO yazmacına aktarılması ile birlikte CRC16 bloğu, kontrol işlemine başlar. Alıcı bloğu CRC kontrol işlemine başladığında, fiziksel katman verinin uzunluğuna bağlı olarak hattan veri almaya devam eder. Hattan alınan verinin uzunluğu, verinin sonu algılanana kadar belli değildir. Kontrol ve yazmaca veri aktarma işlemleri, paralel olarak gerçekleştirildiğinden veri paketinin CRC16 kontrol bitlerini ayıran CRC FIFO yazmacı kullanılır. CRC FIFO yazmacı sayesinde gelen veri 2 byte geciktirilmiş olarak endpoint FIFO yazmacına aktarılır. Veri paketinin sonunu gösteren Rx_Valid kontrol işareti '0' a çekildiği zaman, endpoint FIFO yazmacındaki veriye uygulanan CRC16 işlemi sunucu elde edilen 16 bitlik veri ile CRC FIFO yazmacında tutulan 2 byte'lık veri karşılaştırılır. Karşılaştırılan veriler birbirine eşit ise sırasıyla

senkronizasyon biti, cihazın durumu kontrol edilir.

Hattan alınan verinin uzunluğu 8 byte'ı geçerse, gelen paket değerlendirilmez. Bu durum bir kontrol bayrağı (Data_Size_exceed) ile çıkışa alınır.

Cihaz arabirimi ile cihaz arasında veri aktarmakta arabirim olarak kullanılan FIFO yazmacının yapısı hakkında genel bilgi sunulacaktır.

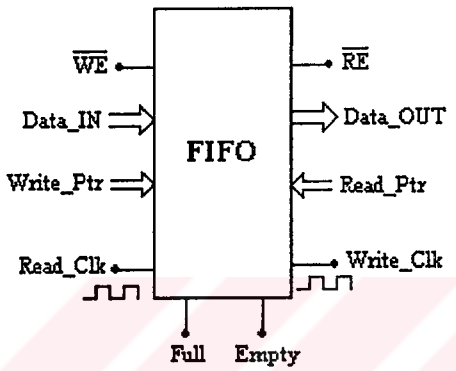


Şekil 6.18 Alıcı ana bloğunda görevli endpoint ve CRC FIFO hafıza hücreleri

6.2.6.1 FIFO (First IN First OUT = İlk Giren İlk Çıkar) Yazmaç Yapısı

FIFO biri yazma, diğeri okumada kullanılan iki porttan oluşur. Adından da anlaşılacağı üzere paralel veri FIFO'ya sırayla yazılır ve yazıldığı sırayla okunur. Yazma ve okuma işleminde, sayıcı çıkışı olan gösterge (pointer) ile bir sonraki yazılacak ve ya okunacak verinin FIFO'daki yeri belirlenir. Okuma sırasında FIFO, hafızasında bulunan mevcut veriyi kaybetmez. FIFO hafıza yapısında, RAM ve ROM'dan farklı olarak hafıza hücrelerine doğrudan erişilmez. Belirli bir veriye ulaşmak için adres bilgisine gerek duyulmadan, göstergeler (pointer) yolu ile hafıza hücrelerine sırasıyla erişilir.

FIFO herhangi bir anda boş ve ya dolu olabilir. Bu durumlara ait bilgi dışarıya kontrol bayrakları ile aktarılır. Yazma ve okuma sırasında bu kontrol bayraklarına ihtiyaç duyulur. FIFO aynı zamanda veri akış hızını da kontrol edebilir. Yüksek hızlı bir veri kaynağından hafıza hücrelerine veri yazılırken, daha düşük hızlı bir cihaz tarafından okunabilir. Böylece FIFO veri akış hızını yüksek hızdan daha düşük hıza çevirmiş olur. Bu da aynı zamanda bir FIFO'nun iki asenkron arabirim arasındaki bağlantıyı sağlaması anlamına gelir. Giriş ve çıkış bağlantı birimleriyle en genel halde FIFO yapısı şekil 6.19'te gösterilmiştir.



Şekil 6.19 FIFO yazmaç yapısı

Proje dahilinde, CRC16 bitlerini ayıran (2 × 8) CRC FIFO yazmacı, USB cihaz arayüzüyle cihaz arasındaki veri aktarımını sağlayan (8 × 8) boyutundaki 2 adet endpoint FIFO yazmacı ve konfigürasyan aşamasında kullanılan (8 × 8) boyutundaki endpoint FIFO yazmacı tasarlanmıştır. Veri aktarımını sağlayan FIFO yazmaçlarından biri alıcı (Receive) ana bloğunda, diğeri gönderici (Transmit) ana bloğunda kullanılmaktadır. CRC FIFO yazmacında 1 bitlik, endpoint FIFO yazmalarında ise 3 bitlik gösterge (pointer) ile bir sonraki yazılacak veya okunacak verinin FIFO'daki yeri belirlenir. FIFO'nun boş olup olmadığını gösteren kontrol bayrağı (FIFO_empty) ile de veri aktarımındaki kontrol sağlanabilir.

6.2.7 OUT İşlem Birimine USB Cihazın Cevap Karakteristiği

OUT token ile seçilen cihaz ve ilgili endpointin FIFO yazmacına, PID ve CRC bitlerinden arındırılmış veri aktarılmadan önce FIFO'nun meşgul olup olmadığı kontrol edilir. Eğer FIFO meşgul ise; aynı verinin ileride tekrar gönderilmesini sağlamak üzere bu durum NAK el sıkışma (handshake) paketi ile hosta bildirilir. FIFO yazmacının meşgul olup olmadığı FIFO_empty kontrol bayrağı ile kontrol edilir. FIFO meşgul değilse, endpoint FIFO yazmacındaki veri CRC16 kontrol işleminden geçirilir. CRC16 kontrol işlemi sonucu veri bozulmuş ise host'a cevap verilmez. Host, zaman aşımı mekanizmasını kullanarak verinin

alıcı tarafa hatalı iletildiğini algılar.

Host, bazı durumlarda herhangi bir uç nokta (kontrol ve Isochronous endpoint dışında) ile veri alışverişini durdurabilir. Host tarafından kilitlenen uç nokta, kontrol bayrağı (HALT) ile bu durumu sisteme bildirir. Bu durumda kilitlenmiş endpointin FIFO yazmacına veri paketi geldiğinde, alınan verinin kontrol işlemleri sonucu doğru alınmış olduğu tespit edilse bile FIFO'daki veri USB cihaza aktarılmaz. İlgili uç noktanın kilitlenmiş olduğu STALL el sıkışma (handshake) paketi ile host'a bildirilir. Eğer kontrol bayrağı (HALT) ile cihazın veri iletimine engel olmadığı belirlenirse, bu sefer cihaz ile host'un veri iletiminde senkron çalışıp çalışmadığı kontrol etmek üzere senkronizasyon bloğu aktif hale getirilir.

Senkronizasyon mekanizması, yığın (bulk), kontrol ve kesme transferler (Isochronous endpoint dışında) için geçerlidir. Host ile cihaz arabiriminin senkron çalışmasını sağlamak üzere gelen veri paketinin PID tipi (DATA0 ve ya DATA1) ile senkronizasyon biti karşılaştırılır. Senkronizasyon biti isochronous uç noktası dışındaki tüm uç noktalarda bulunan 1 bitlik yazmaçtır. Arka arkaya aynı verinin gönderilip gönderilmediğini ve ya iletim esnasında bir veri paketinin kayıp olup olmadığını denetler. PID kod çözücü çıkışında;

- DATA0 PID paket tipi seçili ve SYNC biti '0' değerinde ise
- DATA1 PID paket tipi seçili ve SYNC biti '1' değerinde ise

USB cihaz ile host'un senkron çalıştığı tespit edilir. Bu durumda son kontrol olarak USB cihazın veri almaya uygun olup olmadığının kontrolü yapılır. Eğer PID kod çözücü çıkışında;

- DATA0(1) PID paket tipi seçili ve SYNC biti '1' ('0') değerinde ise,

USB cihaz ile host arasında veri kaybından kaynaklanan uyumsuzluk olduğu belirlenir. Host'a bu durum ACK el sıkışma paketi ile bildirilir. Senkronizasyon mekanizması takip eden bölümde ayrıntılı şekilde anlatılacaktır.

USB cihaz ile host'un senkron çalıştığı tespit edildiğinde USB cihazın veri almaya uygun olup olmadığının kontrolü yapılır. USB cihazdan gönderilen function_ready kontrol bayrağı ile cihazın durumu USB cihaz ara yüzeyine bildirir. Cihaz meşgul ise function_ready kontrol bayrağı lojik '0' seviyesine çekilir. Bu durum host'a NAK handshake paketi ile bildirilir. Cihaz veri almaya uygun ise cihaz tarafından function_ready kontrol bayrağı lojik '1'e çekilir ve host'a ACK el sıkışma paketi gönderilir.

OUT token ile seçilen kesme uç noktasına iletilen veri paketine karşılık host'a cevap vermek üzere yukarıda bahsedilen kontrol işlemlerinden sırasıyla geçilir. OUT işlem birimine USB

cihaz tarafında uygulanan kontrol işlemleri öncelik sırasına göre Çizelge 6.3'te özetlenmiştir.

Çizelge 6.3 OUT işlem birimine USB cihaz tarafında uygulanan kontrol işlemleri

FIFO veri almaya uygun mu?	Alınan veri hatalı mı?	İlgili uç nokta ile veri alışverişi durdurulmuş mu?	Host ile cihaz arasındaki senkronizasyon sağlanıyor mu?	Cihaz veri almaya uygun mu?	Cihazdan gönderilen el sıkışma paketleri
Hayır	E/H	E/H	E/H	E/H	NAK
Evet	Evet	E/H	E/H	E/H	Cevap yok
Evet	Hayır	Evet	E/H	E/H	STALL
Evet	Hayır	Hayır	Hayır	E/H	ACK
Evet	Hayır	Hayır	Evet	Evet	ACK
Evet	Hayır	Hayır	Evet	Hayır	NAK

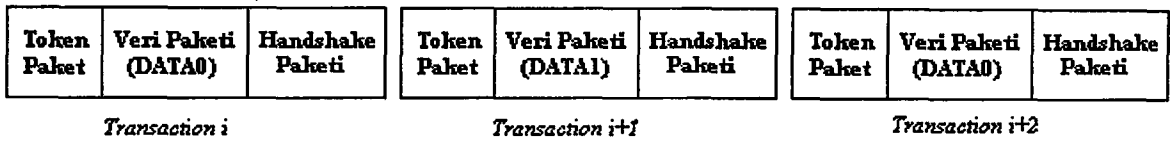
6.2.8 SETUP İşlem Birimine USB Cihazın Cevap Karakteristiği

Setup token paketi ile seçilen kontrol uç noktası konfigürasyon bilgilerini taşıyan setup veri paketini almak zorundadır. Veri paketi alındığına dair host'a ACK el sıkışma paketi ile bildirilir.

6.2.9 Senkronizasyon Kontrol İşlemi

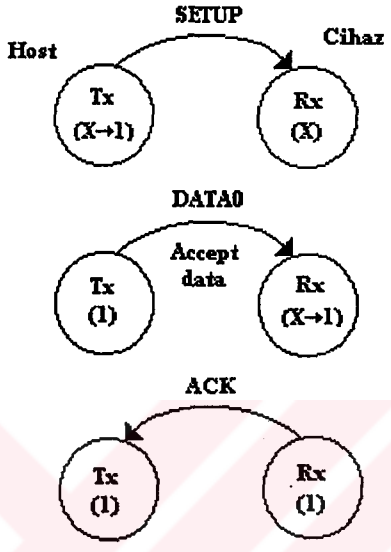
USB sisteminde, alıcı ile gönderici arasında gerçekleşen çoklu işlem birimlerinde arada olabilecek veri kaybını önlemek için senkronizasyon mekanizması kullanılır.

İletilecek veri uzunluğu maksimum paket boyutundan büyük ise, veri USB hattına çoklu işlem birimleri halinde çıkarılır. Ardışıl işlem birimlerinde DATA0 ve DATA1 veri paketi PID tipleri hatta değişimli olarak gönderilir (Şekil 6.20).



Şekil 6.20 USB hattına çıkarılan çoklu işlem birimi yapısı (Compaq, 1998)

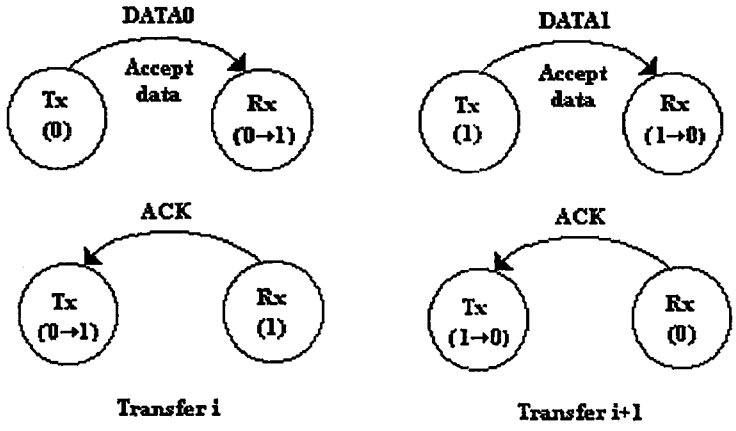
Alıcı ve gönderici arasındaki senkronizasyon, her iki tarafta bulunan 1bitlik senkronizasyon biti yazmacı kullanılarak sağlanır. Başlangıçta her iki tarafta bulunan senkronizasyon bit değerleri birbirine eşittir. Bu eşitlik, konfigürasyon sırasında SETUP işlem biriminin senkronizasyon bit değerlerini lojik '1' seviyesine çekmesi ile sağlanır. (Şekil 6.21)



Şekil 6.21 SETUP işlem birimi ile senkronizasyon bitlerinin başlangıç değerine çekilmesi (Compaq, 1998)

Hattan alınan veri paketi, alıcı bloğunda daha önce bahsedilen kontrol işlemlerinden geçirilip kabul edilmesine engel olmadığı belirlendiğinde gönderici tarafa bu durum ACK el sıkışma paketi ile bildirilir. Alınan veri paketinin PID tipi (DATA0 ve ya DATA1) ile senkronizasyon bitinin uyumlu olduğu tespit edildiğinde alıcı taraftaki senkronizasyon bitinin tümleyeni alınır. Veri paketi gönderen taraf ise, geçerli ACK el sıkışma paketi aldığında, tarafında bulunan senkronizasyon bitinin tümleyenini alır.

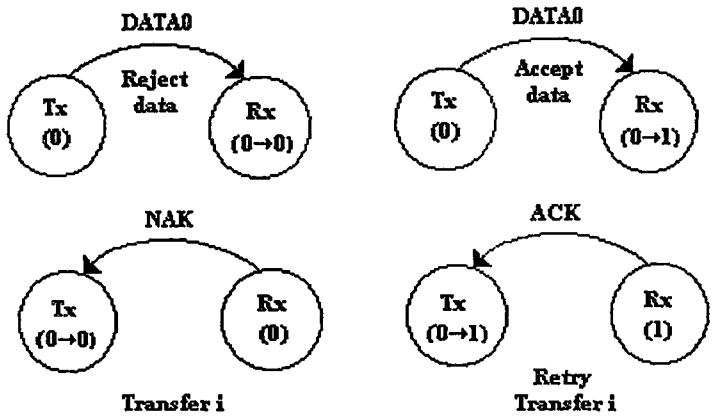
Şekil 6.22'te Ardışıl gerçekleşen iki işlem birimi fazında, alıcı ve gönderici tarafta yer alan senkronizasyon bitlerinin davranışı gösterilmektedir. İlk işlem birimi başlamadan önce her iki tarafın senkronizasyon bitleri lojik '0' değerindedir.



Şekil 6.22 Ardışıl gerçekleşen iki işlem birimi fazında senkronizasyon bitlerinin davranışı

- **Transfer i** :. DATA0 PID tipine sahip veri paketi alıcı tarafta kabul edildiğinde senkronizasyon bitinin tümleyeni alınır (0→1). Alıcı taraf veriyi kabul ettiğine dair ACK el sıkışma paketini veri gönderen tarafa iletir. Veri paketi gönderen taraf ACK paketi aldığında, tarafında bulunan senkronizasyon bitinin tümleyenini alır (0→1). İşlem birimi sona erdiğinde alıcı ve gönderici tarafın senkronizasyon bitleri lojik '1' değerindedir
- **Transfer i+1** : DATA1 PID tipine sahip veri paketi alıcı tarafta kabul edildiğinde senkronizasyon bitinin tümleyeni alınır (1→0). Alıcı taraf veriyi kabul ettiğine dair ACK el sıkışma paketini veri gönderen tarafa iletir. Veri paketi gönderen taraf ACK paketi aldığında, tarafında bulunan senkronizasyon bitinin tümleyenini alır (1→0). İşlem birimi sona erdiğinde alıcı ve gönderici tarafın senkronizasyon bitleri lojik '0' değerindedir.

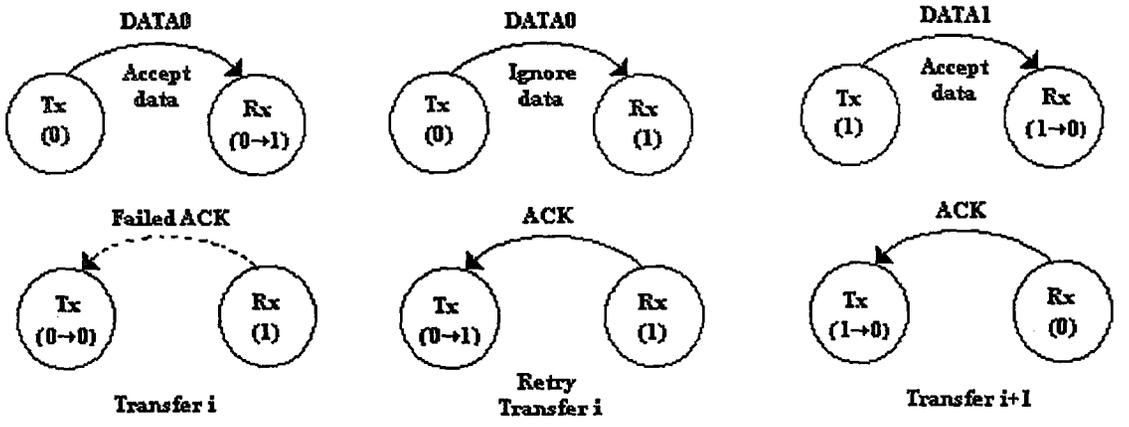
Hattan alınan veri paketinin hatalı olduğu belirlenirse ve ya cihaz veri almaya uygun değilse (meşgul ve ya kilitlenmiş) bu durumda gönderen tarafa ACK el sıkışma paketi iletilmez. Hata koşullarına bağlı olarak veri gönderen tarafa ya hiç veri gönderilmez ya da STALL veya NAK el sıkışma paketlerinden biri gönderilir. Yukarıda belirtilen koşullara bağlı olarak alıcı bloğunda veri kabul edilmediğinden ve gönderici bloğa ACK el sıkışma paketi iletilmediğinden her iki taraftaki senkronizasyon bitleri değerini korur. Alıcı taraf hata durumunda aynı veriyi tekrar iletir. Senkronizasyon bitleri değerini koruduğundan gönderilen veri paketi bir önceki PID veri tipi ile gönderilir. Şekil 6.23'de cihaz meşgulken iletilen veri paketinin her iki tarafta bulunan senkronizasyon bitlerine etkisi gösterilmiştir. İlk işlem birimi başlamadan önce her iki tarafın senkronizasyon bitleri lojik '0' değerindedir.



Şekil 6.23 Cihaz meşgulken iletilen veri paketinin senkronizasyon bitlerine etkisi

- **Transfer i :** DATA0 PID tipine sahip veri paketi alıcı tarafta kabul edilmeden önce belirli kontrol işlemlerinden geçer. Kontroller sonucu USB cihazın meşgul olduğu tespit edilirse bu durum host'a NAK el sıkışma paketi ile bildirilir. Bu durumda alıcı ve veri gönderen taraftaki senkronizasyon bitleri değerini korur (0→0).
- **Transfer i+1 :** Bir süre sonra aynı veri bir önceki PID tipi ile tekrar gönderilir. Kontroller sonucu alınan veri paketinin kabul edilmesinde hiçbir engel olmadığı tespit edilirse, alıcı tarafta bulunan senkronizasyon bitinin tümleyeni alınır (0→1). Alıcı taraf veriyi kabul ettiğine dair ACK el sıkışma paketini veri gönderen tarafa iletir. Veri paketi gönderen taraf, geçerli ACK paketi aldığında, tarafında bulunan senkronizasyon bitinin tümleyenini alır (0→1). İşlem birimi sona erdiğinde alıcı ve gönderici tarafın senkronizasyon bitleri lojik '1' değerindedir.

Hattan alınan veri paketi, alıcı bloğunda kontrol işlemlerinden geçirilip, kabul edilmesine engel olmadığı belirlendiğinde gönderici tarafa bu durum ACK el sıkışma paketi ile bildirilir. Alıcı tarafta verinin kabul edilmesi ile birlikte senkronizasyon bitinin tümleyeni alınır. Fakat veri paketi gönderen tarafa hatalı ulaşan el sıkışma paketi, gönderici tarafta bulunan senkronizasyon bitinin durumunu korumasına ve dolayısıyla alıcı ve gönderici taraftaki senkronizasyonun kaybına neden olur. Veri paketi gönderen taraf hatalı alınan ACK el sıkışma paketine karşılık bir süre sonra aynı veriyi tekrar bir önceki PID tipi ile gönderir. Alıcı tarafta bulunan senkronizasyon biti ile gelen veri paketinin PID verisi uyumlu olmadığından gelen veri paketi değerlendirilmeyecektir ve senkronizasyon biti değerini koruyacaktır. Alıcı taraf gelen veri paketinin, daha önce gelen veri paketi ile aynı olduğunu bildiğinden veri gönderen tarafa ACK paketi gönderir. Veri paketi gönderen taraf geçerli ACK el sıkışma paketini aldığında tarafında bulunan senkronizasyon bitinin tümleyenini alır. Böylece her iki taraf arasındaki senkronizasyon tekrar sağlanmış olur. Şekil 6.24'de Veri paketi gönderen tarafa hatalı iletilen ACK el sıkışma paketinin senkronizasyon bitlerine etkisi gösterilmiştir. İlk işlem birimi başlamadan önce her iki tarafın senkronizasyon bitleri lojik '0' değerindedir. (Compaq, 1998)



Şekil 6.24 Hatalı iletilen ACK el sıkışma paketinin senkronizasyon bitlerine etkisi

- **Transfer i :** DATA0 PID tipine sahip veri paketi alıcı tarafta kabul edildiğinde senkronizasyon bitinin tümleyeni alınır (0→1). Alıcı taraf veriyi kabul ettiğine dair ACK el sıkışma paketini veri paketi gönderen tarafa iletir. Gönderici tarafta hatalı alınan ACK el sıkışma paketi tarafında bulunan senkronizasyon bitinin değerini korumasına neden olur (0→0).
- **Tekrarlanan Transfer i :** Bir süre sonra aynı veri bir önceki PID tipi (DATA0) ile tekrar gönderilir. Alıcı tarafta bulunan senkronizasyon bit değeri lojik '1' ve alınan veri paketinin PID tipi DATA0 olduğundan uyumsuzluk sonucu gelen veri paketi değerlendirilmeyecektir ve alıcı tarafın senkronizasyon bit değeri korunacaktır (1→1). Alıcı taraf gelen veri paketinin, daha önce gelen veri paketi ile aynı olduğunu bildiğinden veri gönderen tarafa ACK paketi gönderir. Veri paketi gönderen taraf geçerli ACK el sıkışma paketini aldığında tarafında bulunan senkronizasyon bitinin tümleyenini alır (0→1).
- **Transfer i+1 :** Yukarıdaki işlemler ile (i+1). transfer başlamadan önce her iki tarafın senkronizasyon bitleri uyumludur. DATA1 PID tipine sahip veri paketi alıcı tarafta kabul edildiğinde senkronizasyon bitinin tümleyeni alınır (1→0). Alıcı taraf veriyi kabul ettiğine dair ACK el sıkışma paketini veri gönderen tarafa iletir. Veri paketi gönderen taraf ACK paketi aldığında, tarafında bulunan senkronizasyon bitinin tümleyenini alır (1→0). İşlem birimi sona erdiğinde alıcı ve gönderici tarafın senkronizasyon bitleri lojik '0' değerindedir.

Bu bilgiler doğrultusunda, USB cihaz arayüzüne ait protokol katmanının tasarımında kesme ve kontrol transferler için senkronizasyon mekanizması gerçekleştirilmiştir. (Compaq, 1998)

6.3 Gönderici (Transmitter) Ana Bloğu

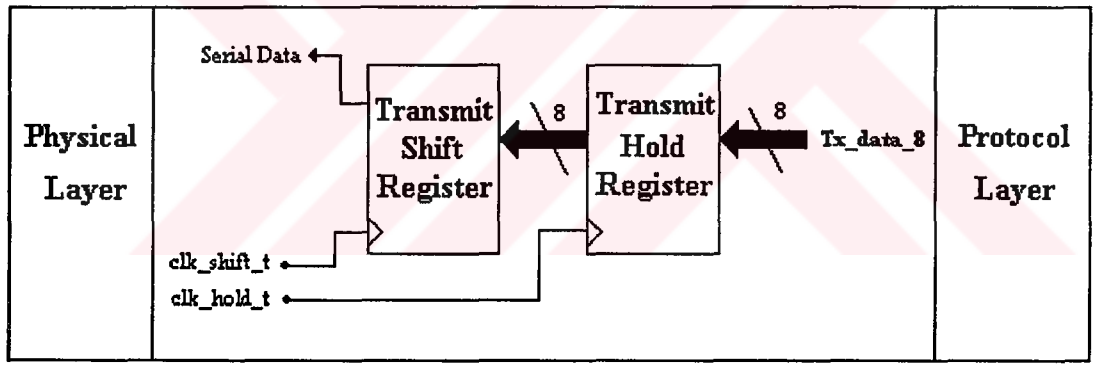
Cihaz tarafından gönderilen ve ya kendi hazırladığı veriyi paket haline getirip fiziksel katman ile arabirim oluşturan arayüz yazmaçlarına iletmekle görevlidir. Veri gönderme sırasında fiziksel katman ile protokol katmanı arasındaki senkronizasyon ve kontrol sağlamak üzere

arayüz yazmaçlarına belirli kontrol işaretleri iletilir. Ayrıca gönderici bloğu ile cihaz arasındaki veri alışverişinde de kontrol işaretleri mevcuttur.

6.3.1 Arayüz Yazmaçlarına Gönderilen Kontrol İşareti ve İletim Hattı

- **Tx_Valid :** Protokol katmanı, ara yazmaçlara veri gönderme işlemine başlamadan önce, USB hattın boş olduğundan emin olmalıdır. Bu kontrol alıcı ana bloğuna iletilen Rx_Active kontrol işareti ile sağlanır. Kontroller sonucu hazırlanan veri paketinin (PID + Veri + CRC bitleri) veya el sıkışma paketinin (PID) 8 bitlik paralel PID verisi hatta yüklendiği anda Tx_Valid kontrol işareti 1'e çekilir. Arayüz yazmaçlarına veri gönderme işlemi tamamlanana kadar kontrol işareti lojik '1' seviyesinde kalır. Fiziksel katmanın son veriyi çekmesi ile birlikte Tx_Valid bayrağı lojik '0' seviyesine çekilir.

- **Tx_data_8 :** Protokol katmanından, fiziksel katmana veri iletimi sırasında arayüzde öteleme yazmacı (shift register) ve tutma yazmacı (hold register) kullanılmaktadır (Şekil 6.25). Bu yazmaçlar protokol katmanından gelen 8 bitlik paralel veriyi, seri veriye çevirip, fiziksel katmana göndermekle görevlidir. Her iki yazmaç 8 bitliktir. Tutma yazmacı protokol katmanından aldığı veriyi gerektiği zaman öteleme yazmacına iletir. Öteleme yazmacı ise, paralel veriyi seriye çevirmek için kullanılır. Protokol katmanı ile tutma yazmacı arasındaki veri iletiminde Tx_data_8 hattı kullanılır.



Şekil 6.25 Fiziksel katmana veri iletmekle görevli ara yazmaçlar

6.3.2 Protokol Katmanına İletilen Kontrol ve Saat İşareti

- **clk_hold_t :** Protokol katmanı çıkışında elde edilen verinin tutma yazmacına yüklenmesinden sorumludur. Belirli bir durumda clk_1x saat işaretinin düşen kenarıyla birlikte 'clk_hold_t' saat işareti 1'e , 'clk_1x' işaretinin diğer düşen kenarında ise clk_hold_r tekrar 0'a çekilir. Arayüz yazmaçları, clk_hold_r saat işareti 1' çekildikten sonra gördüğü 'clk_1x' sinyalinin ilk yükselen kenarında Tx_data_8 hattına yüklenen paralel veriyi çeker.

- **IRQ_device :** USB cihaz tarafından gelen kesme isteği IRQ_device kontrol işareti ile protokol katmanına bildirilir. Cihaz tarafından iletilen verinin aktarılacağı FIFO'nun meşgul olup olmadığı kontrol edilir. FIFO meşgul ise veri USB cihaz tarafında bekletilir. FIFO meşgul değilse cihaz tarafında bulunan veri, clk_1x frekansı ile FIFO'ya aktarılır. FIFO' ya veri yazma işleminde cihaz tarafından protokol katmanına iletilen gösterge (pointer) kontrol işareti ile yazılacak verinin FIFO'daki yeri belirlenir.

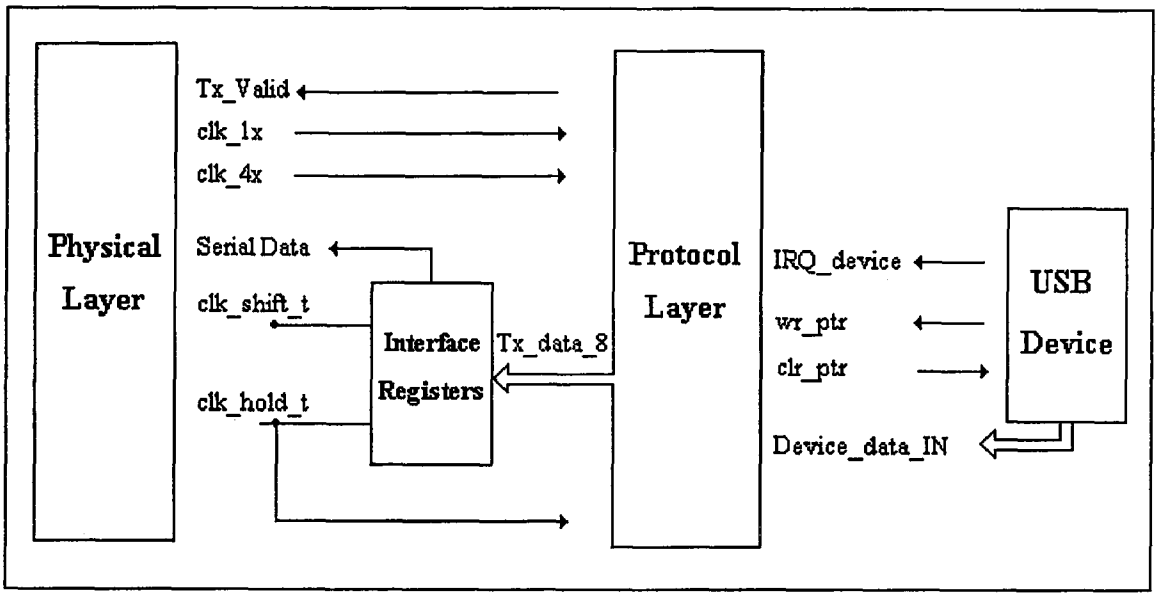
- **Device_data_IN :** USB cihaz kesme isteği ile veri göndermek istediğini protokol katmanına belirtir. FIFO veri almaya uygun ise cihazdaki veri Device_data_IN 8 bitlik paralel veri hattıyla FIFO'ya yüklenir.

- **wr_ptr :** FIFO hafıza elemanının yapısına bağlı olarak yazma işlemi sırasıyla yapılmaktadır. Sayıcı çıkışı olan gösterge (pointer) ile bir sonraki yazılacak verinin FIFO'daki yeri belirlenir. Cihaz tarafından protokol katmanına iletilen gösterge (pointer) kontrol işareti bu işlem için kullanılmaktadır.

6.3.3 USB Cihaza Gönderilen Kontrol İşareti

- **clr_ptr :** Hosta gönderilen veri paketi hatasız alındığında, host tarafından ACK el sıkışma paketi gönderilir. Geçerli ACK paketi USB cihaz ara yüzeyi tarafından alındığında protokol katmanı tarafından 'clr_ptr' kontrol işareti 1'e çekilir. USB cihaz 'clr_ptr' kontrol işaretini lojik '1' değerini algıladığında wr_ptr kontrol girişini sıfırlar. wr_ptr sinyali sıfırlanmadığı sürece FIFO meşgul kabul edilir. Eğer geçerli ACK el sıkışma paketi cihaz arayüzeyine ulaşmaz ise 'clr_ptr' kontrol işareti lojik '1' değerine çekilmez. Dolayısıyla USB cihaz tarafından wr_ptr kontrol girişi sıfırlanamaz. Host tarafından seçilen uç noktaya 'veri gönder!' talebi geldiğinde FIFO da bulunan eski veri tekrar host'a gönderilir. Böylece tasarımda 'Retransmit mekanizması' da desteklenmiş olur.

Fiziksel katman ile protokol katmanı arasında ve USB cihaz ile protokol katmanı arasındaki host'a veri gönderme işleminde kullanılan kontrol işaretleri, saat işaretleri ve veri hatları Şekil 6.26'da gösterilmiştir.



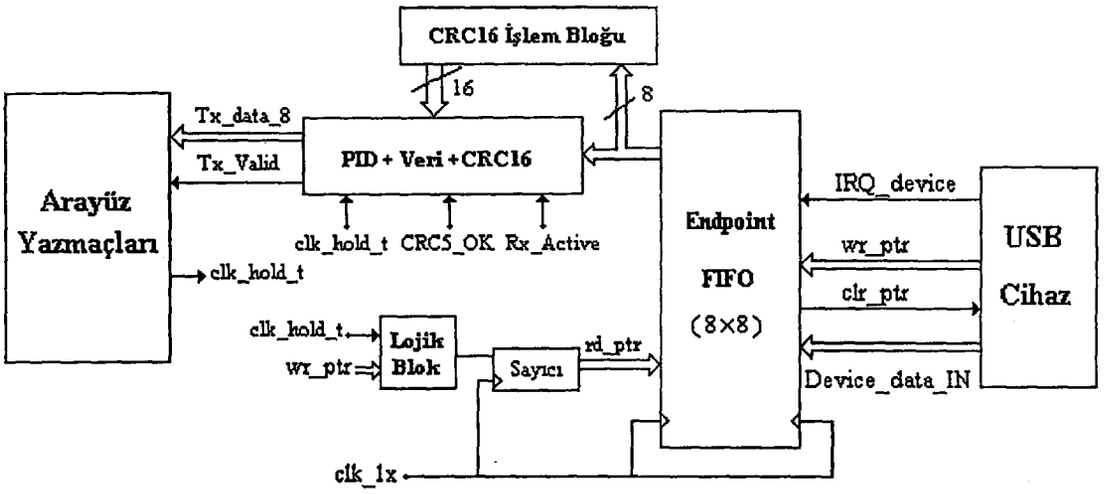
Şekil 6.26 Host'a veri gönderme işleminde kontrolü sağlayan işaretler

6.3.4 IN İşlem Birimine USB Cihazın Cevap Karakteristiği

USB hattından alınan IN token paketi ile seçilen uç noktaya 'veri gönder!' talebi gelir. Gelen token paket alıcı ana bloğunda CRC5 kontrol işleminden geçer. Kontrol sonucunda gelen paketin hatalı alındığı tespit edilirse host'a cevap gönderilmez. Gelen token paketin hatasız alındığı tespit edilirse; ilgili uç noktanın host tarafından kilitlenme durumu kontrol edilir. Kontrol işareti (HALT) ile ilgili uç noktanın kilitlenmiş olduğu belirlendiğinde host'a bu durum STALL el sıkışma paketi ile bildirilir. Protokol katmanı tarafından Tx_Valid bayrağının lojik'1' değerine çekilmesi ile birlikte 8 bitten oluşan PID verisinden oluşan el sıkışma paketi Tx_data_8 veri hattına yüklenir. Arayüz yazmaçları tarafından clk_hold_t saat işareti 1'e çekilir. Arayüz yazmaçları, clk_hold_t saat işaretinin lojik '1' değerine çıktıktan sonra gördüğü 'clk_1x' sinyalinin ilk yükselen kenarında Tx_data_8 hattına yüklenen paralel veriyi çeker. Aynı anda hatta yüklenen verinin sonunu göstermek üzere Tx_Valid kontrol işareti lojik '0' değerine çekilir.

Kontrol işareti (HALT) ile ilgili uç noktanın kilitlenmemiş olduğu belirlendiğinde, cihazdan iletilen veriyi tutmakla görevli FIFO yazmacının durumu kontrol edilir. Host'tan gönderilen veri talebine rağmen, endpoint FIFO yazmacına kesme isteği ile veri yüklenmemişse yani FIFO boş (wr_ptr=0) ise bu durum host'a NAK el sıkışma paketi ile bildirilir.

Kontrol işlemleri sonucu USB cihaz tarafında, host'a veri göndermek için engel olmadığı tespit edilirse, arayüz yazmaçlarına iletmek üzere veri paketi (PID+ Veri +CRC) hazırlanır. Tx_data_8 hattına ilk önce 8 bitlik paralel PID (senkronizasyon bitine bağlı olarak DATA0 ve ya DATA1) verisi çıkarılır. Aynı anda veri paketinin başını gösteren Tx_Valid kontrol işareti protokol katmanı tarafından lojik '1'değerine çekilir. clk_hold_t saat işaretinin lojik '1' değerine çekilmesi ile görülen 'clk_1x' saat işaretinin ilk yükselen kenarında PID verisi ara yazmaçlar tarafından çekilir. Aynı anda FIFO yazmacına ilk yazılan veri Tx_data_8 hattına çıkarılır. Veri Tx_data_8 hattına yazılırken aynı zamanda CRC16 bitleri üretmek üzere clk_4x frekansı ile alıcı bloğunda bulunan CRC16 işleminden geçirilir. Bu işlem endpoint FIFO yazmacında bulunan tüm veri hatta yazıldığı sürece tekrarlanır. Verinin ardından alıcı bloğunda üretilen CRC16 bitleri hatta çıkarılır. Ara yüz yazmacının, veri paketindeki son 8 bitlik paralel veriyi (CRC16 [15:8]) çekmesi ile protokol katmanının veri gönderme işlemi sona erer. Veri gönderme işleminin bittiğini ara yüz yazmaçlarına belirtmek üzere Tx_Valid kontrol işaretini '0' a çeker.



Şekil 6.27 USB cihazdan gönderilen verinin FIFO hafıza birimi aracılıyla hatta çıkarılması

IN işlem birimine USB cihaz ara yüzeyinin protokol katmanında uygulanan kontrol işlemleri öncelik sırasına göre Çizelge 6.4'te özetlenmiştir.

Çizelge 6.4 IN işlem birimine USB cihaz tarafında uygulanan kontrol işlemleri

Alınan token Paketi hatalı mı?	İlgili uç nokta ile veri alışverişi durdurulmuş mu?	FIFO veri göndermeye uygun mu?	Protokol katmanı tarafından gönderilen paket tipleri
Evet	E/H	E/H	Cevap yok
Hayır	Evet	E/H	STALL
Hayır	Hayır	Hayır	NAK
Hayır	Hayır	Evet	Veri paketi

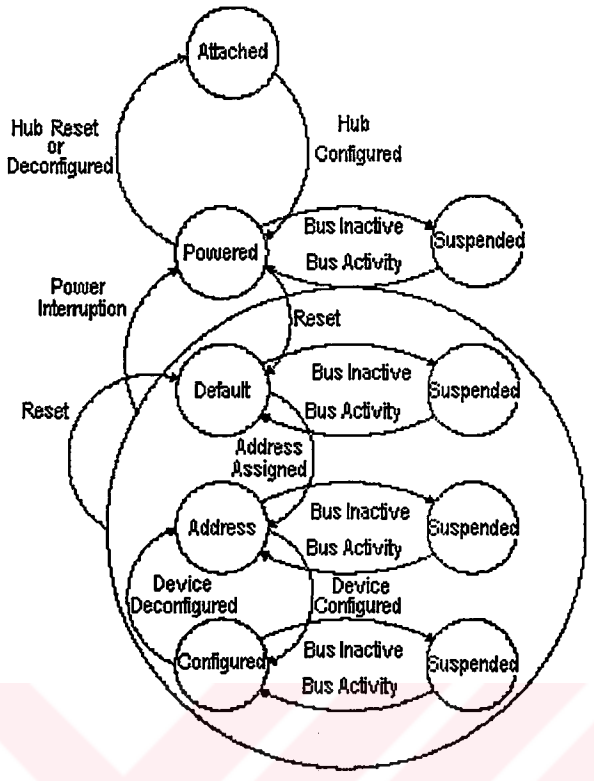
6.4 Kontrol Ana Bloğu

USB sistemine, dışarıdan USB cihaz bağlandığında, hub vasıtasıyla sistemdeki değişiklik algılanır. Hub, portlarında bulunan durum bilgileri kontrol ederek, portlarından birine cihaz bağlandığını ve bağlanan cihazın hızını algılayabilme özelliğine sahiptir. USB cihaz, sisteme bağlandığında, cihaz ile host arasındaki veri alış verişi gerçekleşene kadar sisteme yeni bağlanan cihaz belirli evrelerden geçer. Şekil 6.28'de gösterilen durum makinesi ile bu evreler arasındaki geçişler özetlenmiştir.

Cihaz sisteme bağlandığında USB hattan ve / veya kendi gerilim kaynaklarından (self powered) beslenebilir. Cihaz, besleme kaynağını hangi uç birimden sağladığına dair bilgiyi tanımlayıcılar vasıtasıyla host'a bildirir. USB cihaza güç verildikten sonra, cihaz hattan RESET işareti alana kadar, host tarafından iletilen hiçbir transfere cevap vermez. RESET sinyali ile USB cihaz bünyesinde bulunduran tüm yazmaçları belirli bir başlangıç değerine koşullar. Bu durumda, cihaza host tarafından adres atanana kadar USB cihaz varsayılan adrese (default address=7'b0000000) iletilen işlem birimlerine cevap verecektir. Cihaza, host tarafından adres atanması ile birlikte cihaz varsayılan durumdan, adreslemiş duruma (default state → address state) geçecektir ve adres değerini, hatta RESET sinyalini görene kadar koruyacaktır. Bu durumda cihaz konfigürasyon işlemine başlamaya hazırdır. Konfigürasyon, cihaz ve cihazı oluşturan alt birimlerin (configuration, interface, endpoint) karakteristiklerini kontrol transfer vasıtasıyla host'a iletme işlemidir. Ayrıca bu işlem ile cihaza ait bazı özellikler host tarafından belirli bir değere koşullandırılabilir. Konfigürasyon sırasında, setup işlem birimi ile host tarafından konfigürasyon değeri (config_value) atanır. Böylece cihaz, adreslemiş durumdan, konfigüre edilmiş duruma geçer (address state → configured state).

Durum makinesinden görüldüğü gibi (Şekil 6.28) cihaza güç verildikten sonra, USB cihaz, hatta 3ms boyunca herhangi bir trafik görmediğinde otomatik olarak bekleme (suspend) durumuna geçer. Bekleme durumundayken 500μA'den fazla akım tüketilmez. Bu durumda cihaz kendi iç durumunu korur (adres, konfigürasyon bilgisi vb.). Hatta veri trafiğinin olmaması host'un kendisinin de bekleme durumuna geçmiş olmasından kaynaklanabilir. Hatta veri transferi meydana geldiğinde USB cihaz bekleme durumundan çıkar. Ayrıca uzaktan uyandırma (remote wakeup) özelliğini destekleyen cihazlar, host'u elektriksel sinyalleşme ile bekleme durumundan çıkarabilirler.

USB cihaz arayüzü tasarımında sisteme bağlanan cihazın uzaktan uyandırma özelliği desteklenmemiştir. Cihazı bekleme durumundan sadece host çıkarabilir.



Şekil 6.28 SETUP işleminde USB cihazın davranışını özetleyen durum diyagramı (Compaq, 1998)

Konfigürasyon sırasında cihazın karakteristiklerini host'a göndermek ve cihazın bazı özelliklerinin belirli başlangıç değere koşullandırılmasını sağlamak üzere standart istek paketleri tanımlanır. Standart cihazlara ilişkin istekler USB cihaza kontrol transferi vasıtasıyla iletilir. Bütün USB cihazlar standart istek paketlerini desteklemek zorundadır. Standart olmayan, üreticiye (vendor) ve ya sınıfa özgü tasarlanmış cihazlarda standart cihaz isteklerine ek olarak özel istek paketleri tanımlanır. Tasarımda standart cihazları destekleyen USB cihaz arayüzü öngörüldüğünden konfigürasyon sırasında standart cihazlara özgü istek paketlerine ait karakteristikler sunulacaktır.

6.4.1 Standart Cihaz İstek Paketleri

Standart istekler ve bunlara ilişkin parametreler kontrol transfer vasıtasıyla, SETUP işlem birimine ait veri paketi içinde gönderilir. 4. bölümde de bahsedildiği gibi kontrol transfer setup fazı, veri fazı ve durum (status) fazı olmak üzere 3 fazdan oluşabilir. Setup fazında cihaza iletilen veri paketi ile standart istek paketinin tipi, hangi birime iletildiği (cihaz, interface, endpoint), veri fazında iletilecek veri paketinin uzunluğu ve yönü belirlenir. Bazı istekler sadece bu 8 byte'lık bölüm tarafından tamamen belirlenebildiğinden onlar için ayrıca veri fazına gerek yoktur. Diğer istekler için ise 8 byte'dan daha fazla verinin okunmasına ve ya yazılmasına ihtiyaç vardır. Bu durumda veri fazı gereklidir. Kontrol transferinde veri

fazının bulunup bulunmaması istek paketinin tipine göre değişir. Setup işlem birimine ait veri yapısı Şekil 6.29'da gösterilmiştir.

1byte	1byte	2 bytes	2 bytes	2 bytes
bmRequest Type	bRequest	wValue	wIndex	wLenght

Şekil 6.29 Setup işlem birimine(Transaction) ait veri yapısı (Compaq, 1998)

bmRequest Type : Veri fazında iletilecek verinin yönü, cihazın tipi ve istek paketinin yönlendirilmiş olduğu birim 1 byte'lık veri içinde cihaza gönderilir.

D_7 : Veri transferinin yönü	$D_{6..5}$ Cihaz tipi	D4..0 Alıcı birim
0 : Host'tan cihaza	2'b00: Standart	4'b0000 : Cihaz
1 : Cihazdan host'a	2'b01: Class	4'b0001 : Interface
	2'b10: Vendor	4'b0010 : Endpoint

bRequest : Standart cihaz istek paketinin tipini belirler. (Clear_Feature, Get_Status vb..)

wValue ve wIndex : İçerdiği bilgi istek paketlerinin tipine göre değişir.

wLenght : Kontrol transferin veri fazında iletilecek verinin uzunluğunu belirtir. Veri uzunluğu '0' değerinde ise kontrol transferi setup ve durum (status) fazı olmak üzere 2 fazda tamamlanır.

İstek paketleri, ilgili USB cihaz tarafından alındığında, paket içinde yer alan parametreler ile istek tipi uyumlu değilse veya mevcut olmayan bir özelliğe veya o anda aktif olan konfigürasyona ait olmayan arayüz veya uç noktalara gönderildiğinde istek hatası oluşur. Hata, host'a STALL el sıkışma paketi ile iletilir.

6.4.1.1 Standart Cihaz İstek Tipleri

Setup işlem biriminin veri paketinde, standart cihaz isteklerine özgü bilgiler iletilir. Veri paketinde kontrol transferin veri fazında iletilecek verinin uzunluğunu belirten bilgi bulunur. Bu bilgiye bağlı olarak kontrol transferi 2 ve ya 3 fazda tamamlanabilir.

Get_Status : Talep edilen birime bağlı olarak USB cihaz, arayüz ve ya uç noktalara ait durum bilgilerini host'a gönderilir. Hedef birim cihaz ise, uzaktan uyandırma özelliğini destekleyip

desteklemediğini, besleme kaynağını hangi birimden aldığını (self powered or bus powered), hedef birim olan uç nokta ise veri alışverişinin durdurulmuş (HALT) olup olmadığı host'a bildirir.

Clear_Feature : İstek paketi alan hedef birim cihaz ise, daha önce Set_feature istek paketi ile aktif edilen uzaktan uyandırma özelliği clear_feature istek paketi ile etkin olmayan (disable) hale getirilir. Hedef birim uç nokta ise, daha önce Set_feature istek paketi ile kilitlenmiş uç nokta, clear_feature istek paketi ile normal veri alış-verişine devam eder.

Set_Feature : İstek paketi alan hedef birim cihaz ise, uzaktan uyandırma özelliği aktif hale getirilir. Hedef birim uç nokta ise, ilgili uç noktanın veri alış verişi durdurulur.

Set_Address : USB cihaza adres ataması, Set_Address istek paketi ile gerçekleşir. Cihaz atanan adres değerini reset alana kadar ve ya güç kesilinceye kadar koruyacaktır.

Get_Descriptor : Cihaz ve cihaza ait alt birimlerin (konfigürasyon, arayüz ve endpoint) karakteristiklerini hosta iletimini başlatır. Get_Descriptor komutu ile cihaz, konfigürasyon ve ya string tanımlayıcılara ait bilgilere ulaşılır. String tanımlayıcılar ile, kullanıcının okuyabileceği biçimde tanımlayıcılar görüntülenir. String tanımlayıcılara ait bilgileri kullanmak opsiyonel olduğundan, USB arayüz tasarımında, bu birime ait istekler desteklenmemiştir. Get_Descriptor komutu ile hedef birim cihaz tanımlayıcıları ise 18byte'tan oluşan cihazın karakteristiklerini belirleyen veriler gönderici ana bloğunda hazırlanıp hatta çıkarılacaktır. Hedef birim konfigürasyon tanımlayıcıları ise, ilk önce konfigürasyon tanımlayıcılarında bulunan bilgiler, ardından konfigürasyona ait arayüz bilgileri ve arayüze ait endpoinlere ilişkin tanımlayıcı bilgileri hosta iletilir. Tasarımda herhangi bir düşük hızlı cihaza ait konfigürasyon, konfigürasyona ait bir arayüz ve arayüze ait 3 adet uç nokta desteklenmiştir.

Set_Descriptor : Bu istek paketi varolan tanımlayıcıları güncellemek veya yenilerini eklemek için kullanılabilir. Tasarımda bu talep desteklenmediğinden, host'tan Set_descriptor istek paketi geldiğinde, USB arayüzü bu paketin tanımlı olmadığını STALL el sıkışma paketi ile host'a bildirecektir.

Get_Configuration : USB cihaz sahip olduğu konfigürasyon değerini (config_value), bu istek paketi ile host'a gönderir. İletilen değer '0' ise, USB cihaz konfigure edilmemiş demektir.

Set_ Configuration : Cihaza istek paketi içinde iletilen konfigürasyon değeri, ya konfigürasyon tanımlayıcısında bulunan değer (config_value) ile eşleştirilmeli yada sıfır değerinde olmalıdır.

Get_Interface : Bu istek paketi arayüze özgü, seçilen değişimli ayarları host'a bildirmeyi sağlar. Tasarımda kullanılan arayüz değişimli ayarları kullanmadığından, host'a, arayüz tanımlayıcılarında belirtilen default değeri (00h) gönderilir.

Set_Interface : Arayüze özgü değişimli ayarlar, bu istek paketi ile, host tarafında seçilir. Tasarımda arayüze özgü değişimli ayarlar için varsayılan ilk değer (default) desteklendiğinden, bu istek paketine USB arayüz tarafından STALL handshake paketi ile cevap verilir.

Synch_Frame : USB cihaz, isochronous transferi destekliyorsa, host ile cihaz arasındaki senkronizasyon host'tan 1ms aralıklar ile gönderilen SOF paketindeki çerçeve numarası takip edilerek sağlanır. Bu istek paketi ile çerçeve numarası ve senkronizasyona ait bilgiler host'a bildirilir. Tasarımda isochronous transfer desteklenmediğinden, host'tan Synch_Frame istek paketi geldiğinde, USB arayüz STALL el sıkışma paketi ile host'a cevap verir.

6.4.2 Standart USB Tanımlayıcıları

USB cihazlar, sahip oldukları nitelikleri tanımlayıcıları vasıtasıyla host'a bildirirler. Tasarımda USB cihaz, konfigürasyon, arayüz (interface), uç nokta (endpoint) olmak üzere 4 farklı tanımlayıcı kullanılmıştır. Hiyerarşik düzen içinde bulunan tanımlayıcılar, host'tan gelen istek paketindeki bilgiler doğrultusunda, host tarafından talep edilen uzunluğa bağlı olarak bir veya birkaç işlem birimi ile gönderici ana bloğu tarafından USB hatta çıkarılırlar. Her tanımlayıcı kendisinin byte olarak toplam uzunluğunu bildiren 1 byte'lık alana sahiptir. Ayrıca tüm tanımlayıcılara, tanımlayıcı tipini (cihaz, konfigürasyon, arayüz (interface), uç nokta (endpoint)) belirten 1 byte'lık alan ayrılmıştır. Diğer bilgiler tanımlayıcılara özgüdür. Takip eden bölümde standart tanımlayıcılara ait genel bilgiler verilecektir.

6.4.2.1 Cihaz Tanımlayıcısı

Cihaz Tanımlayıcısı, USB cihaza ait genel özellikleri tanımlanır. Herhangi bir USB cihaz bir adet cihaz tanımlayıcısına sahiptir. Konfigürasyon aşamasında kullanılan kontrol transfere ait haberleşme kanalının boyutu (maksimum paket uzunluğu), sahip olduğu konfigürasyon sayısı, üretime özgü bilgileri kapsayan ürün kodu, üretici kodu, cihaz seri numarası vb. bilgiler cihaz tanımlayıcılarında yer alır.

6.4.2.2 Konfigürasyon Tanımlayıcısı

Konfigürasyon tanımlayıcısı, USB cihaza özgü konfigürasyon bilgilerini içerir. Bir USB cihaz birden fazla konfigürasyon tanımlayıcısına sahip olabilir. Konfigürasyon değeri, konfigurasyon tarafından desteklenen arayüz(interface) sayısı, konfigürasyon karakteristikleri (besleme kaynağı ve uzaktan uyandırma) ve USB hattından çekilen maksimum akım bilgisi konfigürasyon tanımlayıcılarında yer alır. Host'tan konfigürasyon tanımlayıcılarına istek paketi geldiğinde, talep edilen veri uzunluğuna bağlı olarak konfigürasyonun desteklediği tüm arayüz tanımlayıcılarına ve uç nokta tanımlayıcılarına ait bilgiler hiyararşik düzende gönderici ana bloğu tarafından USB hatta çıkarılır. Bu nedenle konfigürasyon tanımlayıcısında tüm alt tanımlayıcıların (arayüz ve endpoint) toplam veri uzunluğunu veren 2 byte'lık veri alanı mevcuttur.

6.4.2.3 Arayüz (Interface) Tanımlayıcısı

Arayüz tanımlayıcısı, konfigürasyon içinde yer alan belirli bir arayüze ait bilgileri içerir. Bir konfigürasyon, her biri konfigürasyon içindeki bir grup uç noktayı (endpoint) temsil eden bir veya daha fazla arayüzü destekleyebilir. Arayüz tanımlayıcısı standart istek paketleri ile doğrudan erişilemez. Daha önce de bahsedildiği gibi host'tan Get_Descriptor istek paketi geldiğinde konfigürasyon tanımlayıcısının ardından USB hatta çıkarılır.

6.4.2.4 Uç nokta (Endpoint) Tanımlayıcısı

USB cihazda bulunan her bir arayüz, bir veya birden fazla uç noktaya sahip olabilir. Arayüzde kullanılan her uç nokta kendi özel tanımlayıcısına sahiptir. Uç nokta adresi, desteklediği transfer tipi (Kontrol, Yığın, Kesme, Isochronous), periyodik transferler için host tarafından seçme aralığı, haberleşmede kullanılacak maksimum veri uzunluğu bilgileri uç nokta tanımlayıcılarında yer alır. Uç nokta tanımlayıcısı da arayüz tanımlayıcısı gibi host tarafından standart istek paketleri ile doğruda erişilemez. Arayüz tanımlayıcısının ardından USB hatta çıkarılır.

6.4.3 Sayım İşlemi (Bus Enumeration)

USB cihaz, sistemde bulunan hub'ın herhangi portuna takıldığında ve ya çıkarıldığında host, cihaz durum değişiklilerini belirlemek ve idare etmek için sayım işlemi olarak bilinen bir proses gerçekleştirir. USB cihaz, güç verilmiş bir hub portuna takıldığında aşağıdaki işlemler sırasıyla gerçekleşir.

- USB cihaz, hub portuna dinamik takılıp ve çıkarılma özelliğine sahiptir. Hub, portuna bağlanan herhangi bir USB cihazı port durum yazmaçlarının çıkışlarını oluşturan kontrol işaretleri aracılığıyla hosta bildirir. Bu anda cihaz güç verilmiş durumdadır (powered state) ve bağlı olduğu port erişime açık değildir (disabled).

- Host, hub'tan aldığı kontrol işaretlerini değerlendirerek, sistemdeki değişikliğin nedenini belirler.

- Değerlendirmeler sonucunda host, yeni bağlanan cihazın, hub'ın hangi portuna takıldığını bilmektedir. Host, hatta yeni bir cihaz bağlanma sürecinin tamamlanması ve cihazın güç seviyesinin kararlı hale gelmesini sağlamak için en az 100ms boyunca bekler.

- Host daha sonra ilgili porta, port erişim (port enable) ve reset komutu gönderir. İlgili hub bu reset işaretini portta 10ms boyunca tutar ve 10ms sonunda port artık erişime açıktır (enabled). Bu anda USB cihaz, varsayılan (default) duruma geçer ve hattan en fazla 100mA akım çeker. Reset sinyali ile birlikte USB cihazın tüm yazmaçları belirli başlangıç değere koşullandırılır. Cihaz bu durumda varsayılan adrese (default address) gönderilen komutlara cevap verir.

- Host, cihaza (varsayılan adrese) Get_Device_Descriptor standart istek paketi yollar ve 18 byte'lık cihaz tanımlayıcısını okur. Host, cihaz tanımlayıcısını okuyarak, konfigürasyon sırasında kullanılacak varsayılan kanalın (default pipe) maksimum boyutunu belirler.

- Host, Set_Address komutuyla, USB cihaza reset alana kadar kullanacağı, yegane bir adres atar, böylece cihaz adreslenmiş duruma (adressed) geçer.

- Host yeni atadığı cihaz adresine tekrar Get_Device_Descriptor komutu yollar ve cihaz tanımlayıcısını tekrar okur. Bu iki işlem sırasında elde edilen verilerin aynı olması gerekir. Herhangi bir farklı değer veya zaman aşımı durumu bir hata olduğunu gösterir ve host tarafından sistem yazılımı müdahalesi gerektirir.

- Daha sonra host, Get_Configuration_Descriptor komutuyla USB cihazın konfigürasyon bilgilerini konfigürasyon tanımlayıcısından, konfigürasyon sayısına bağlı olarak sırasıyla okur. Bu sürecin tamamlanması birkaç ms sürebilir.

- Alınan konfigürasyon bilgilerine ve USB cihazın ne şekilde kullanılacağına bağlı olarak host, cihaza bir konfigürasyon değeri atar (configuration value). Bu anda cihaz artık konfigüre edilmiş durumdadır (configured state). USB cihaz, konfigürasyon tanımlayıcısında belirtilen miktarda hattan akım çeker. USB cihaz, artık fonksiyonel kullanıma hazırdır.

SONUÇLAR

Bu çalışmada tasarımı hedeflenen USB cihaz arayüzü tasarım projesinin bir bölümünü oluşturan protokol katmanına ait bloklar kod düzeyinde tasarlanmıştır. Tasarımda USB 1.1 spesifikasyonunun gerektirdiği tanımlar temel alınmıştır.

Düşük hızda (Low speed : 1.5 Mb/s) veri alış verişini sağlayan ve tasarlanan modülün cihaz tarafındaki arayüz ile uyumlu kontrol girişlerine sahip herhangi bir cihaz tasarlanan sisteme bağlandığında USB cihaz gibi davranır. Ayrıca tasarımda orta hızlı cihazların desteklenebilirliği de sağlanmıştır. Tasarım için Verilog HDL tanımlama dili kullanılmış ve oluşturulan kodlar Synopsys Design-Analyzer programı ile sentezlenmiştir. Kod düzeyinde tasarlanan modüllerin, gecikmeler dikkate alınarak kapı düzeyinde eşdeğer devreleri elde edilmiştir. Sistemin doğruluğu Cadence DFII tasarım ortamının temel lojik simülatörü Verilog XL kullanılarak, hazırlanan test modülleri ile sınanmıştır. Elde edilen simülasyon sonuçları ile USB 1.1 protokolünün gerektirdiği kriterlerin uyumlu olduğu gözlenmiştir. Tasarlanan modül FPGA ortamına aktarılmış ve test kartı üzerinde Logic Tester (LV500) ve bir PC yardımıyla sistem gerçek ortamda test edilmektedir.

KAYNAKLAR

Anderson D., (1999), USB System Architecture, Addison-Wesley, Canada.

Compaq, (1998), Universal Serial Specification Revision 1.1, USA

Hyde J., (1999), USB Design by Example, Intel University Press, USA.

Synopsys Customer Education Services (1998), Chip Synthesis Workshop Student Guide, USA.

Thomas D.E., (1994), The Verilog Hardware Description Language, Kluwer Academic Publisher, USA

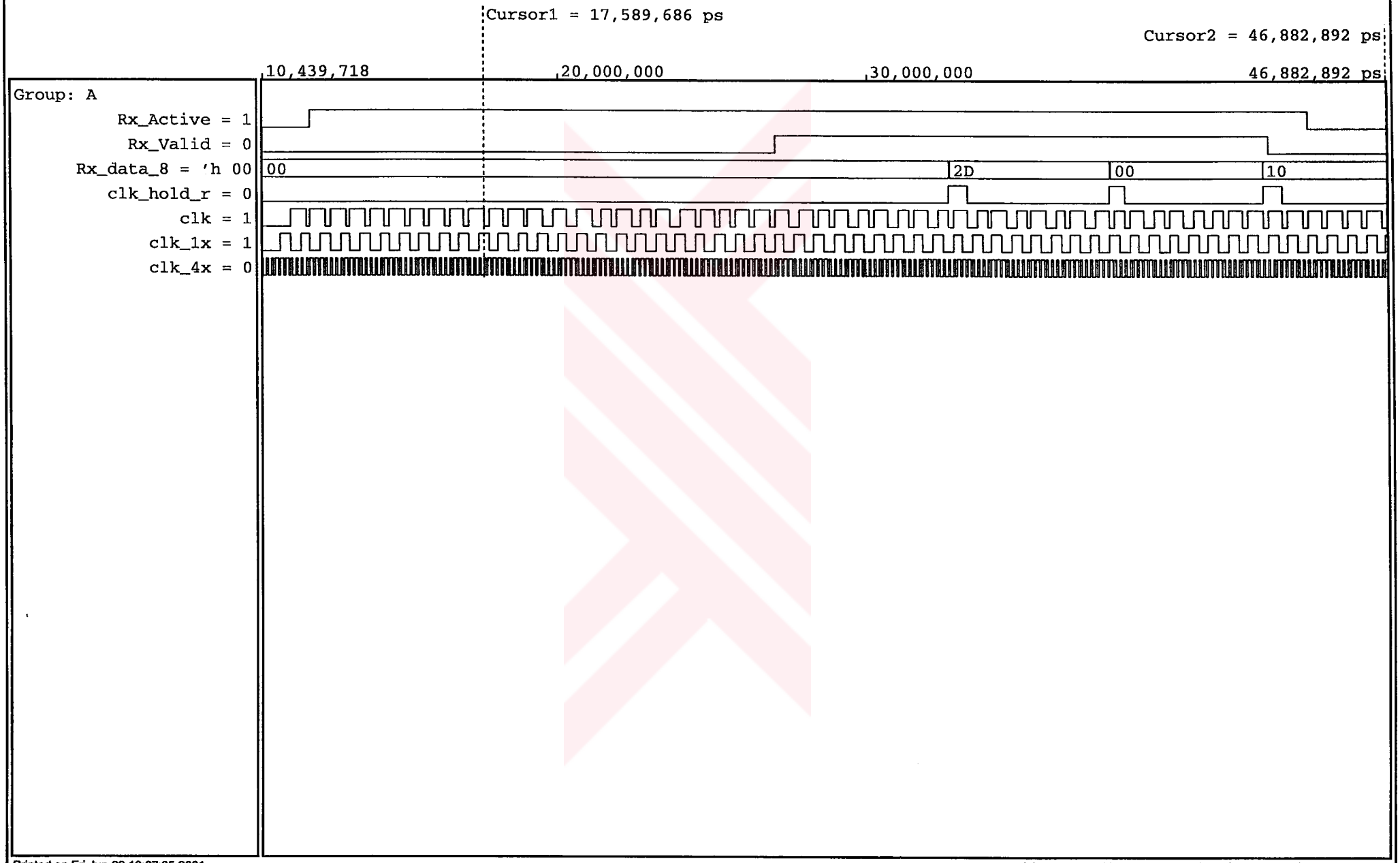
Tocci .R. J. (1991), Digital Systems Principles and Applications, Prentice Hall, USA.

USB Tasarım Grubu, (2001), USB Arayüzü Tasarım Raporu1, ETA ASIC Tasarım Merkezi, İstanbul

EKLER

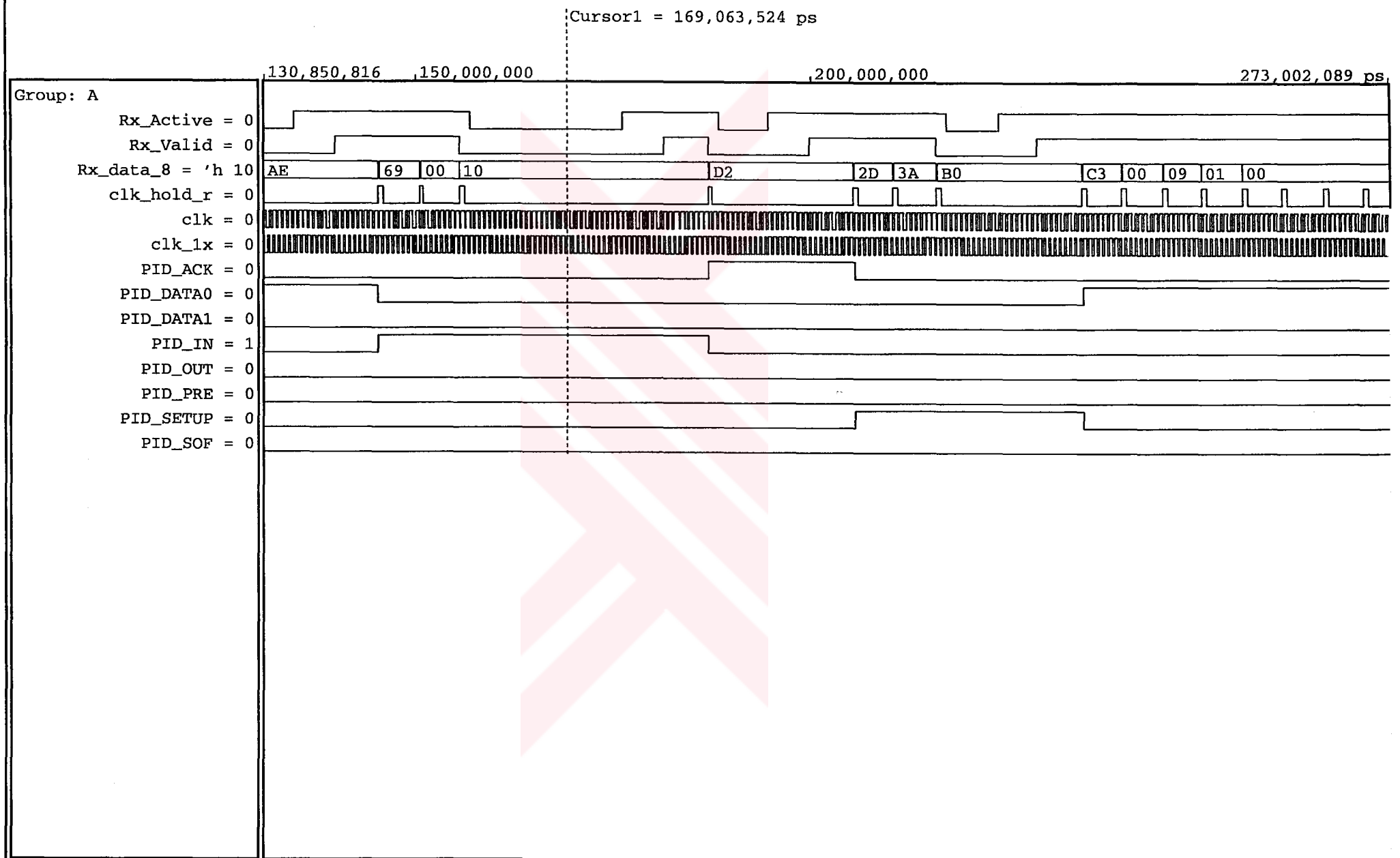
ITU ETA ASIC TASARIM MERKEZI

Arayuz Yazmaclarindan Protokol Katmanina Iletilen Kontrol ve Saat Isareti



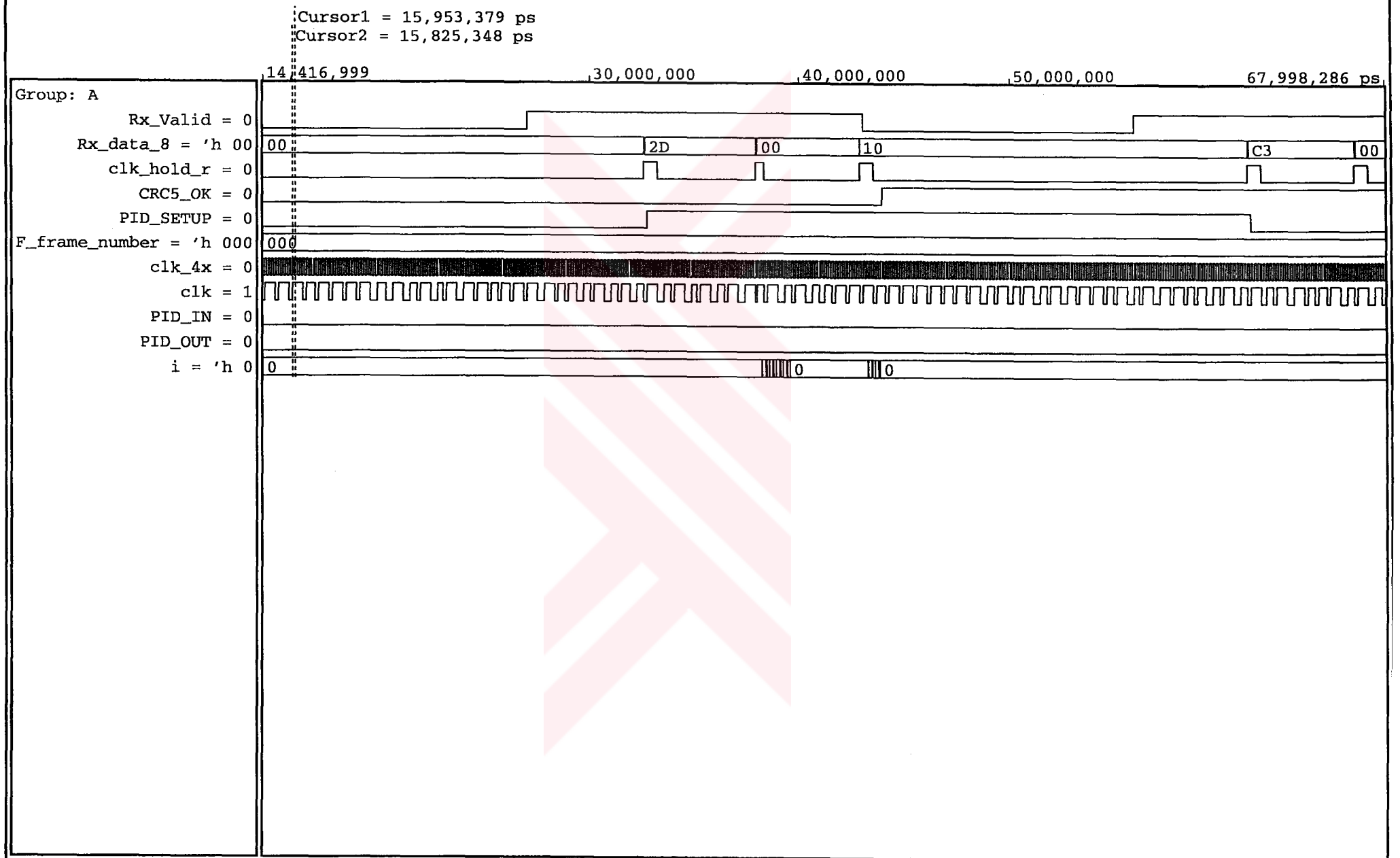
ITU ETA ASIC TASARIM MERKEZI

PID Kod Cozucu Cikisinda Elde Edilen PID Tipleri



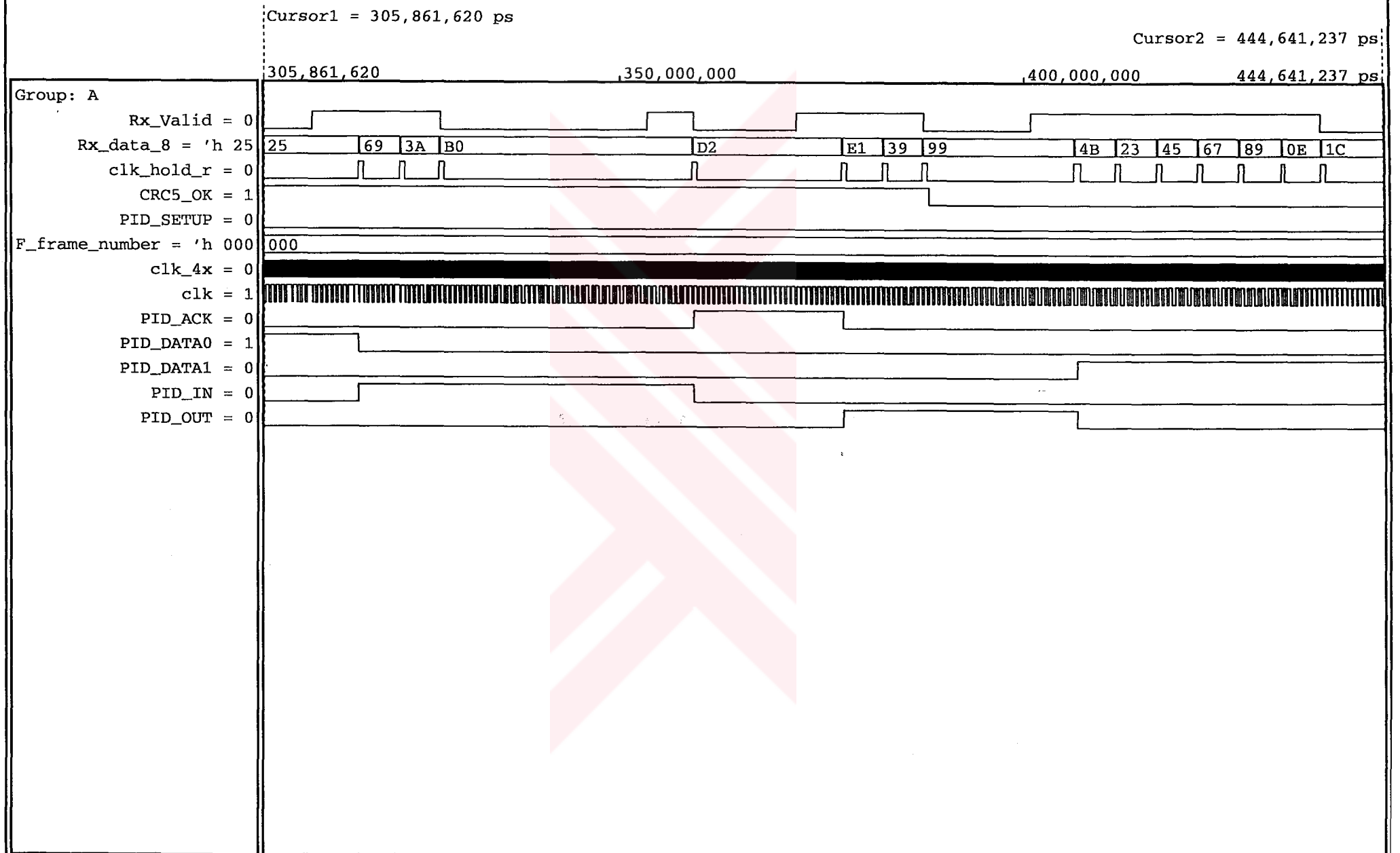
ITU ETA ASIC TASARIM MERKEZI

SETUP Token Paketine Uygulanan CRC5 Kontrol Islemi



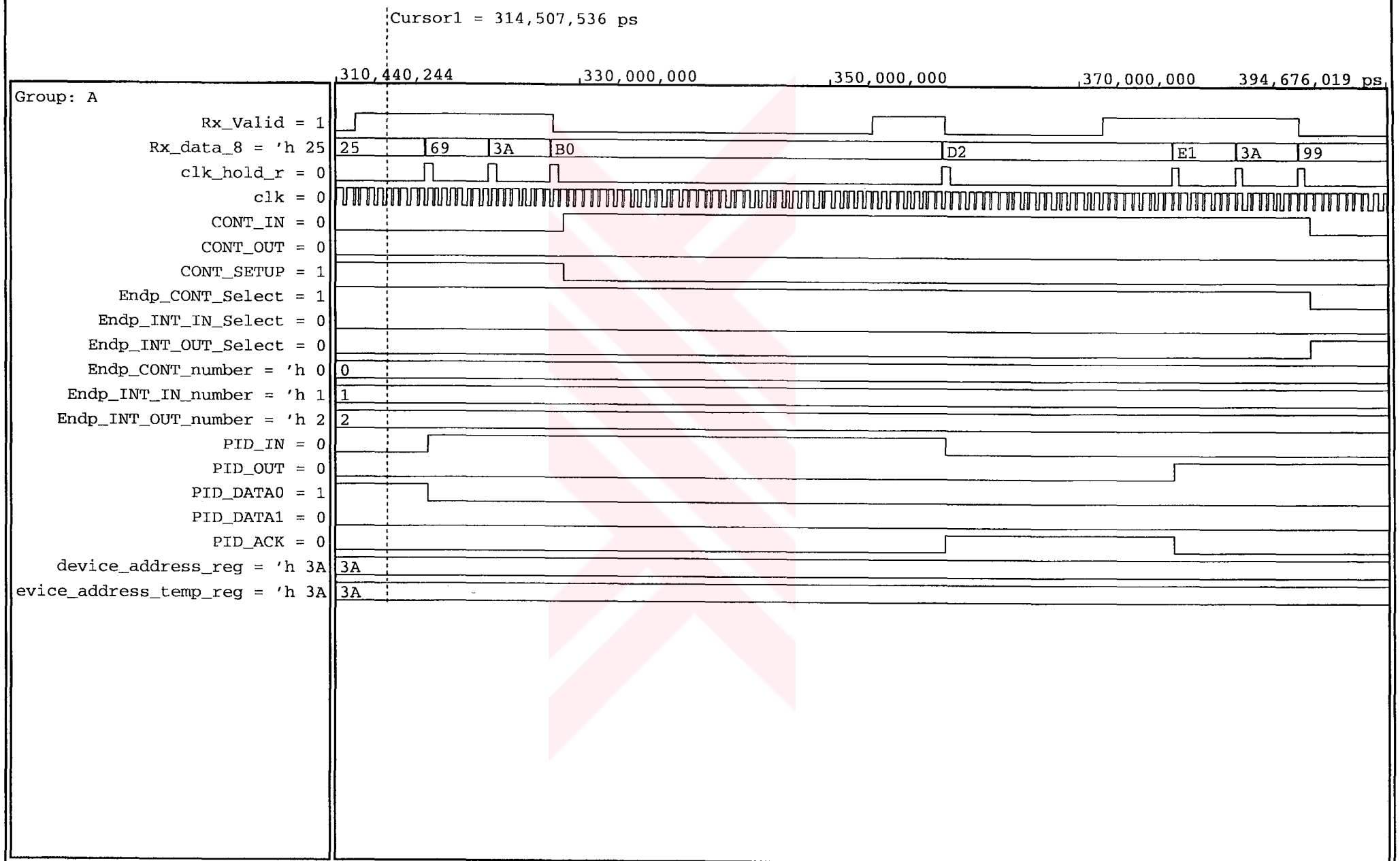
ITU ETA ASIC TASARIM MERKEZI

OUT Token Paketinin Hatali Elde Edilmesi



ITU ETA ASIC TASARIM MERKEZI

Endpoint Adresi ile Cihaz Adresinin Token Paket Icinden Elde Edilmesi

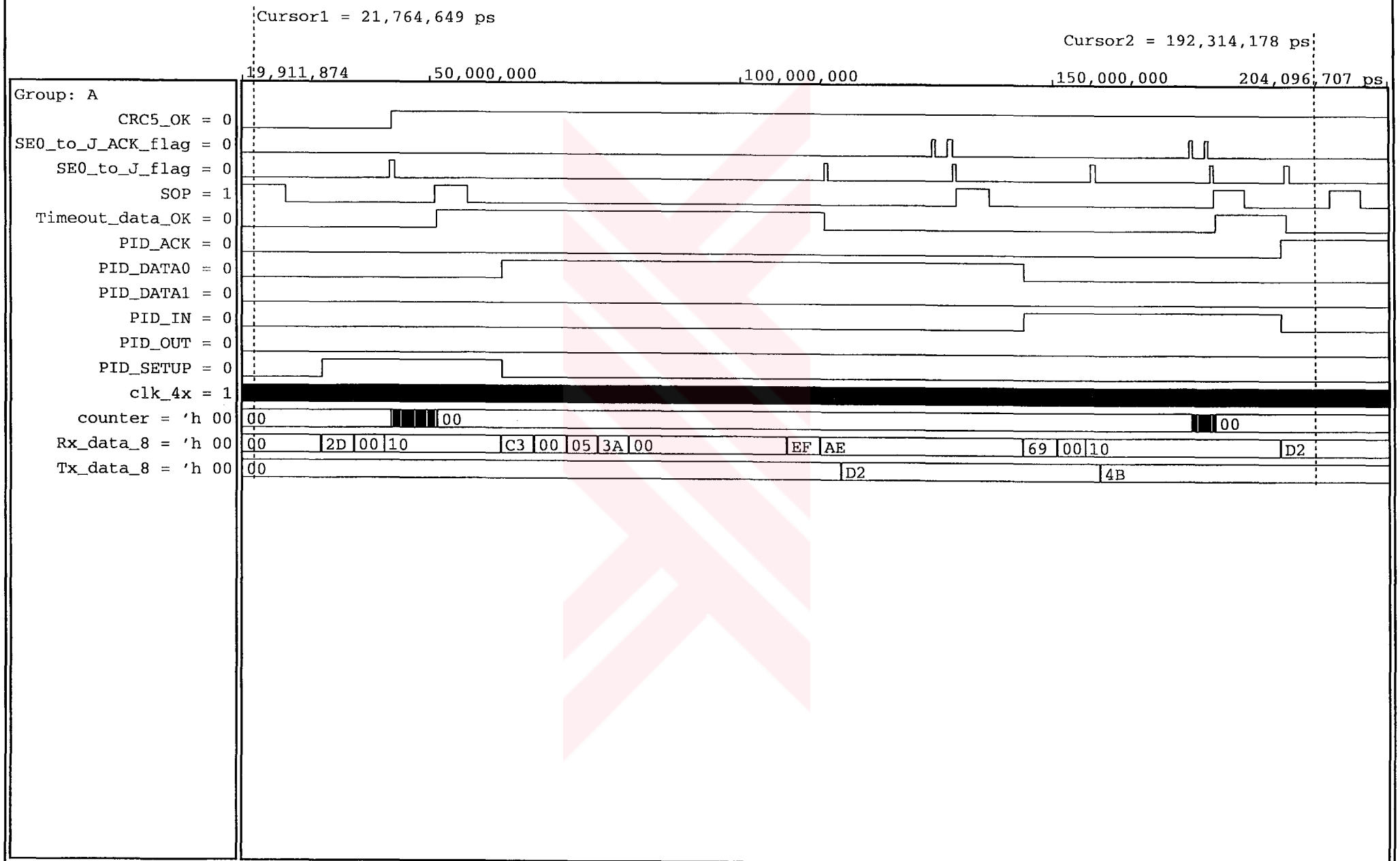


Printed on Fri Jun 22 11:13:52 2001

Printed by DAI Signalscan 6.3s2 from Cadence Design Systems, Inc.

ITU ETA ASIC TASARIM MERKEZI

Timeout Kontrol Islemi

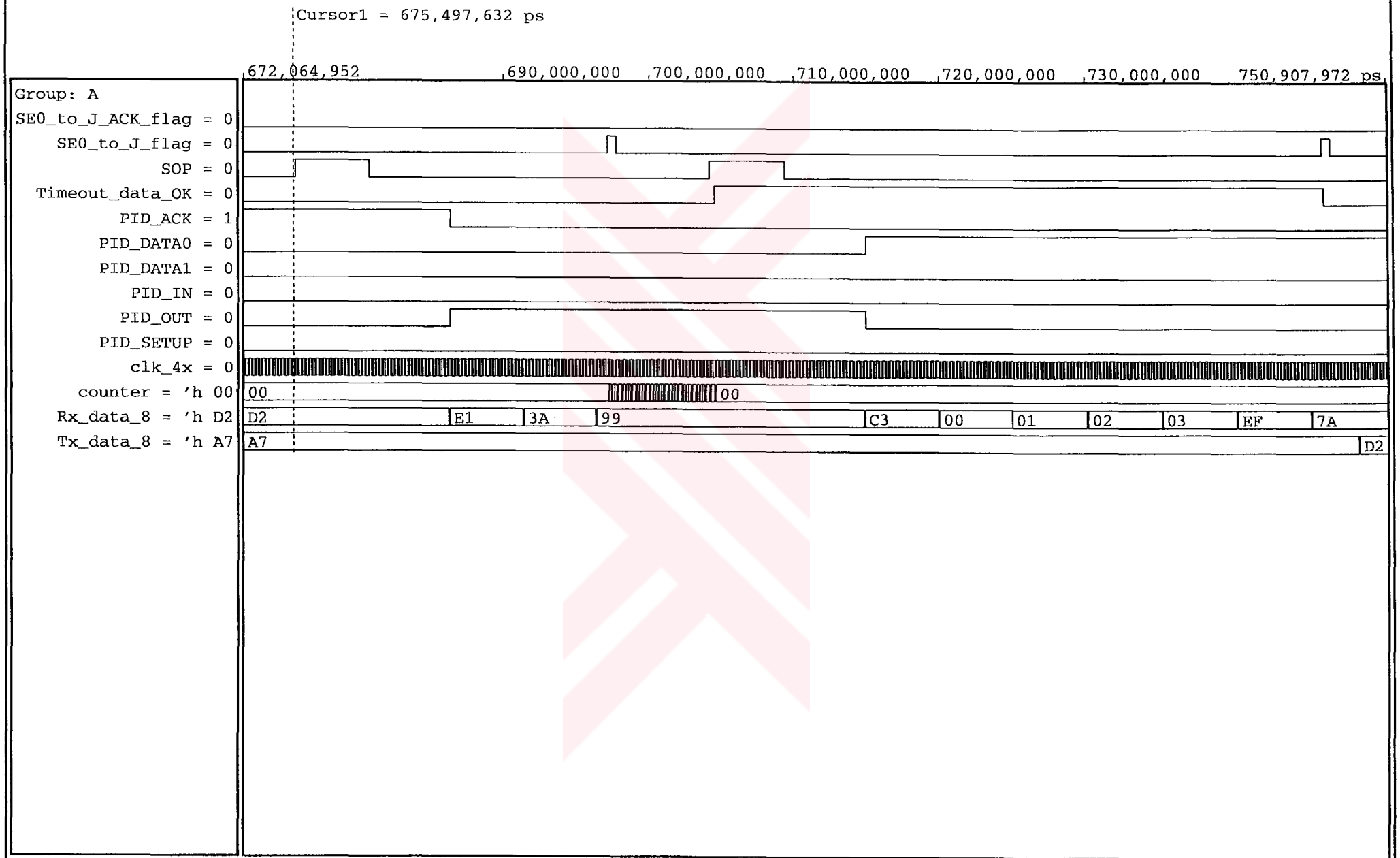


Printed on Fri Jun 22 11:24:40 2001

Printed by DAI Signalscan 6.3s2 from Cadence Design Systems, Inc.

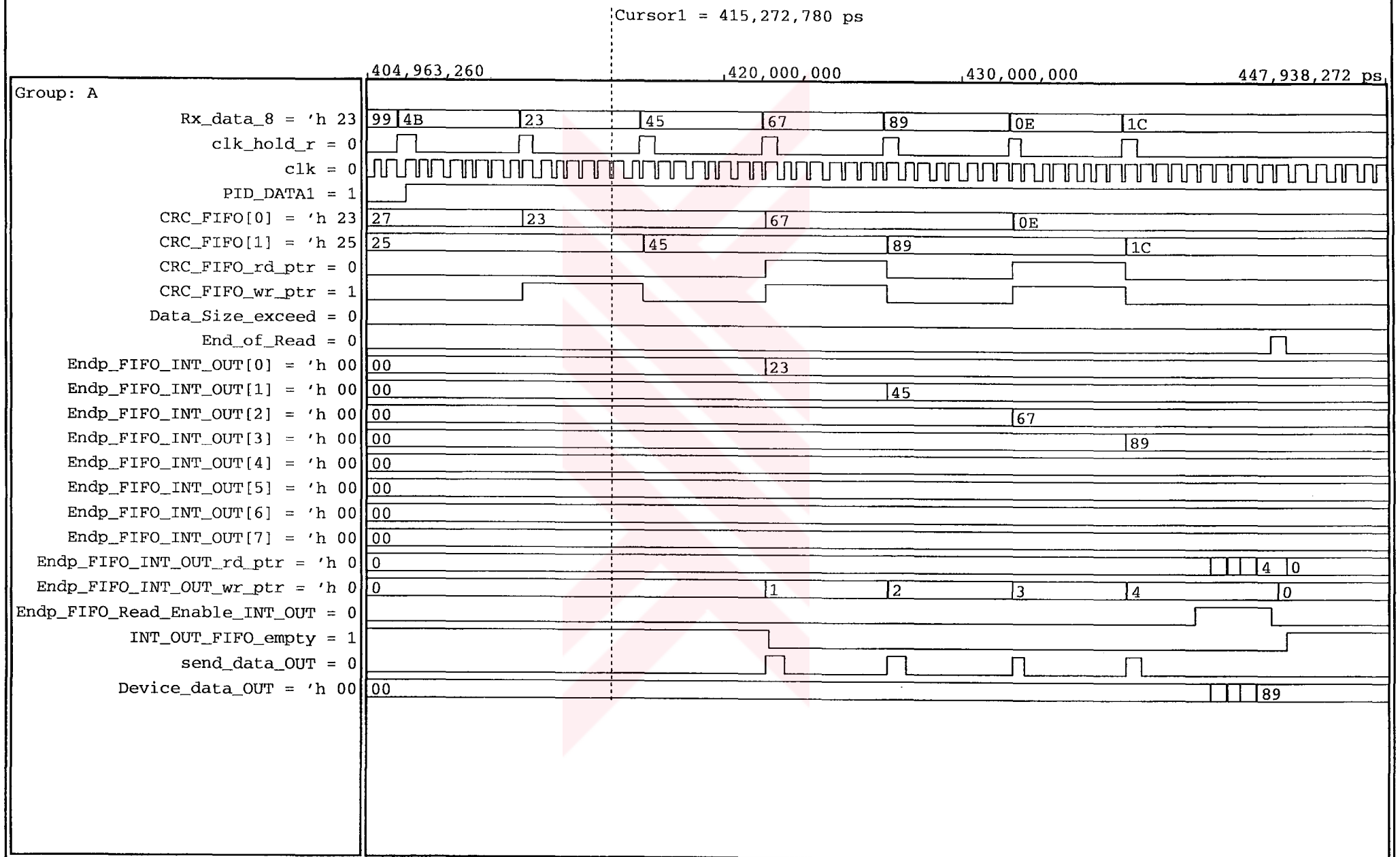
ITU ETA ASIC TASARIM MERKEZI

SETUP Paketinin Ardindan Timeout Suresi Icinde Gelen Data0 paket



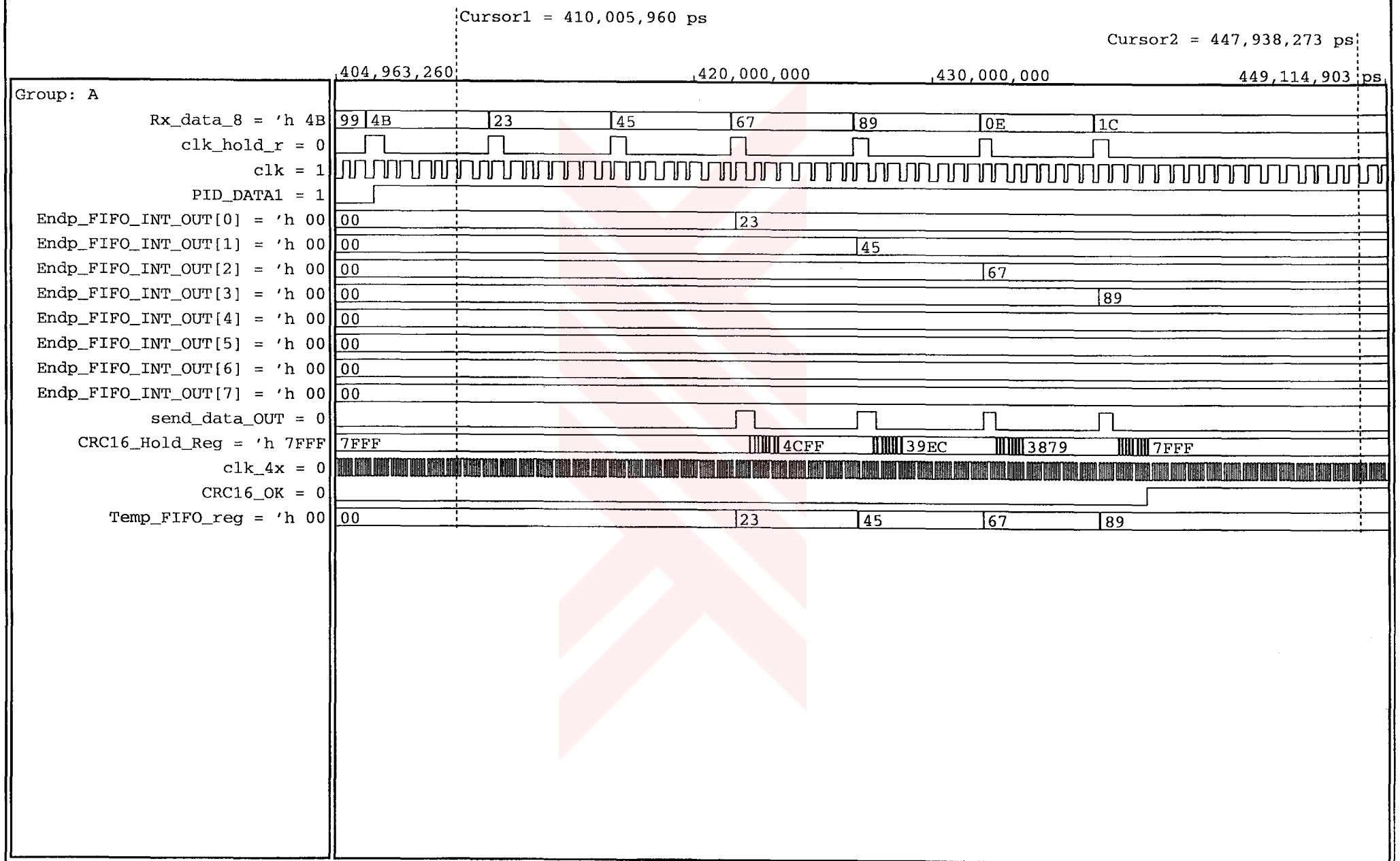
ITU ETA ASIC TASARIM MERKEZI

Veri Hattindan Alinan Verinin CRC FIFO'ya ve Endpoint FIFO'ya Yazilmasi



ITU ETA ASIC TASARIM MERKEZI

Endpoint FIFO'da Bulunan Veriye Uygulanan CRC16 Kontrol Islemi

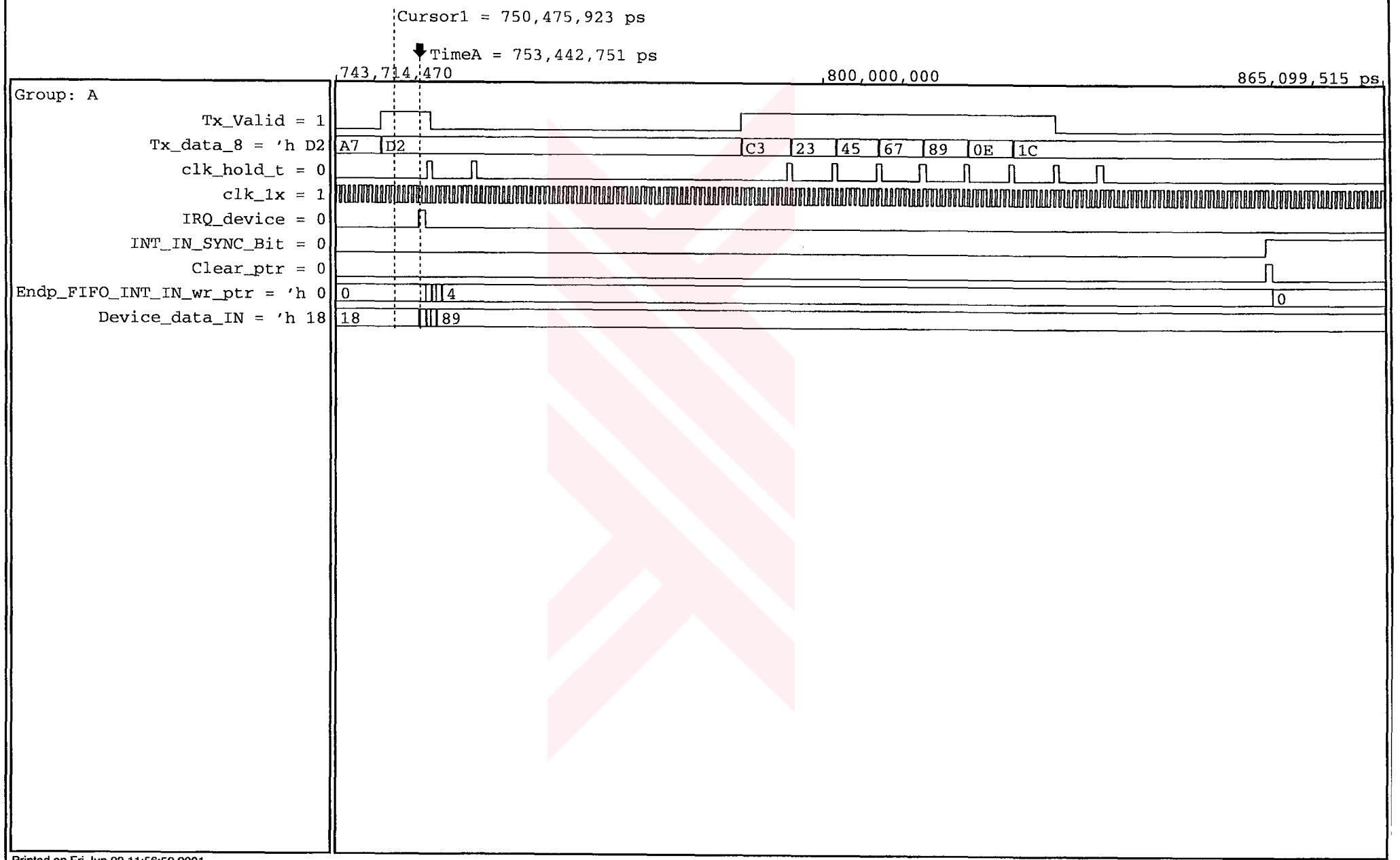


Printed on Fri Jun 22 11:46:32 2001

Printed by DAI Signalscan 6.3s2 from Cadence Design Systems, Inc.

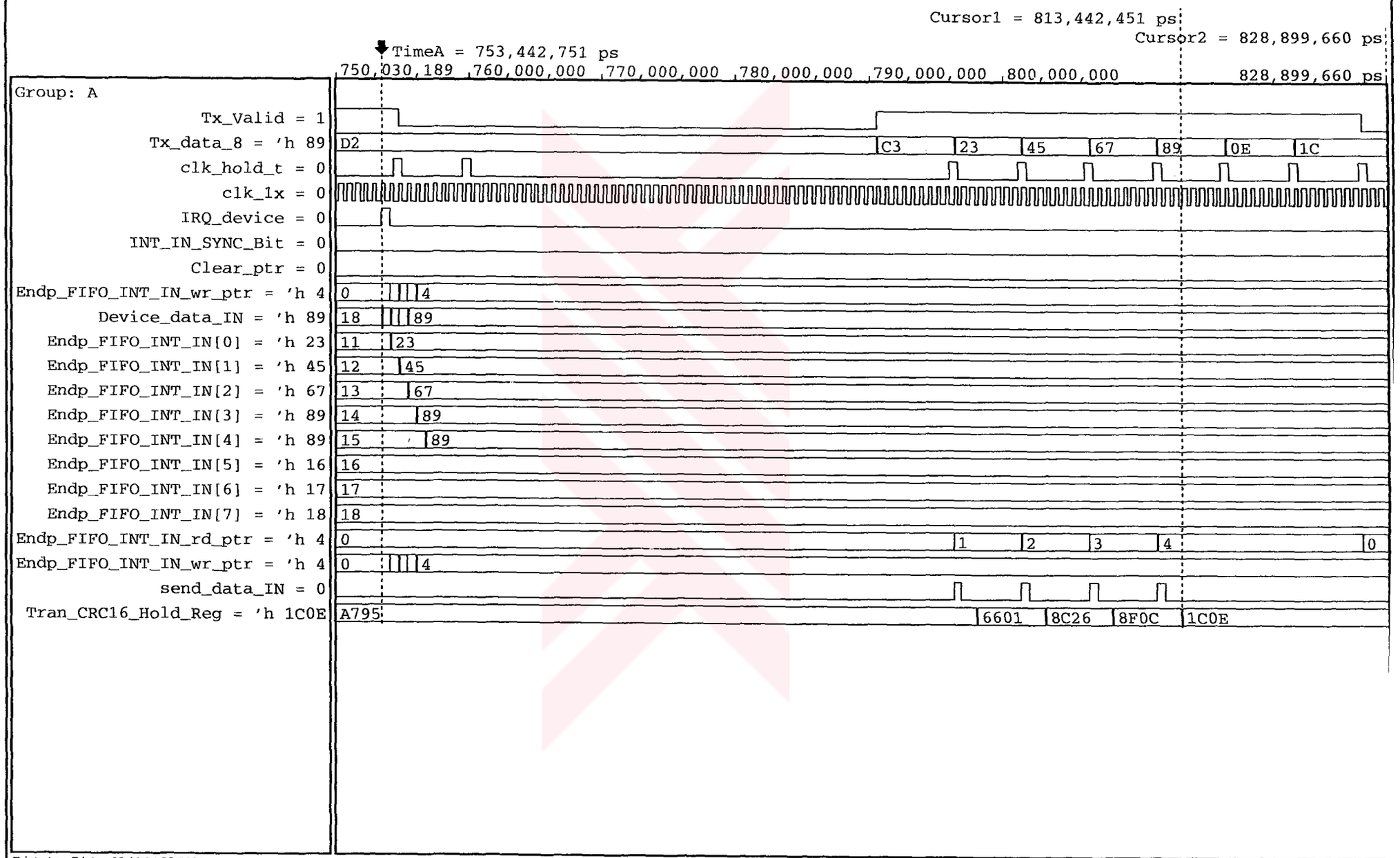
ITU ETA ASIC TASARIM MERKEZI

Host'a Veri Gondermekle Gorevli Kontrol ve Saat Isatretleri



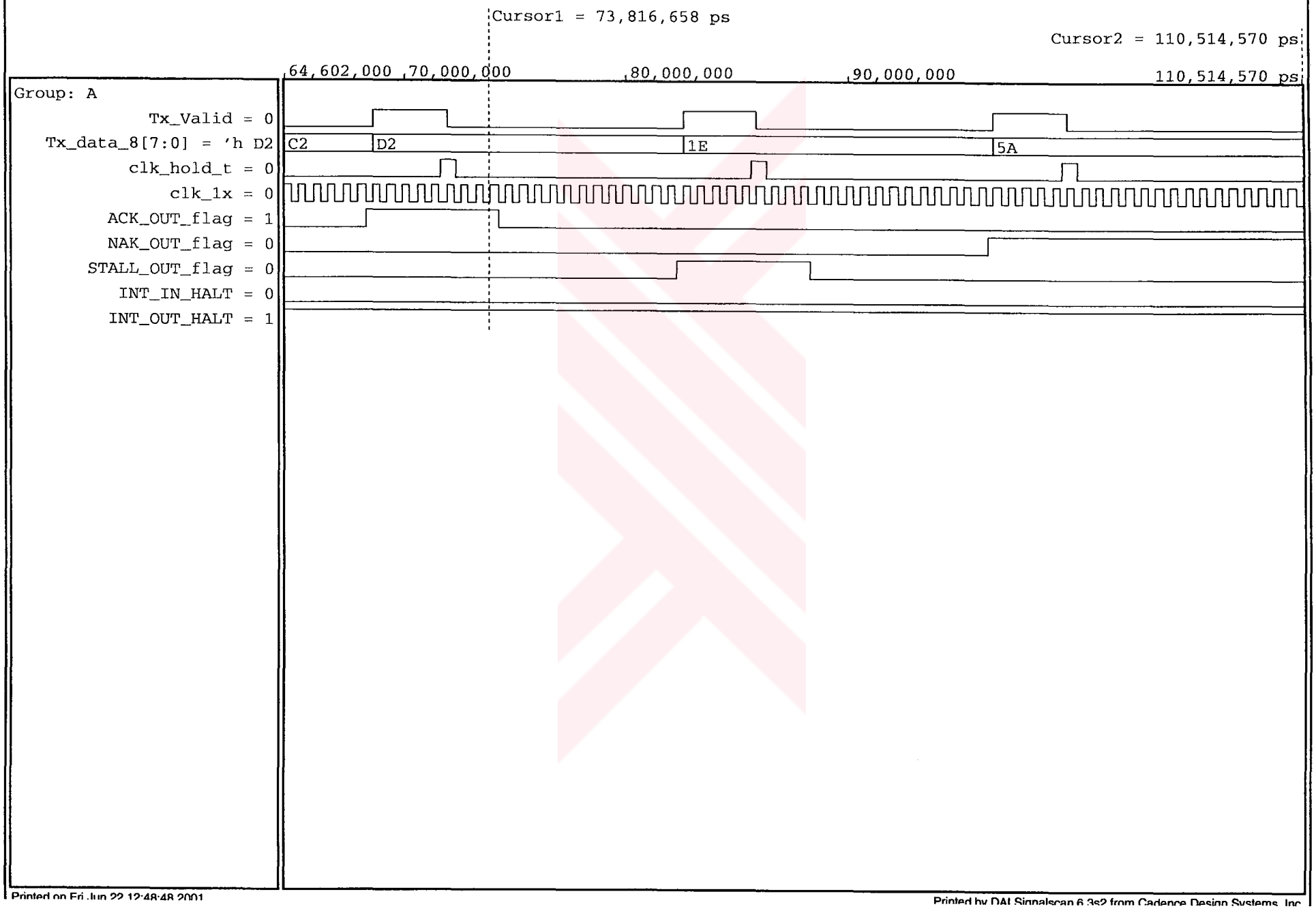
ITU ETA ASIC TASARIM MERKEZI

USB Cihazdan Gonderilen Verinin FIFO Yazmacina Yuklenmesi ve CRC16 Bitlerinin Uretilmesi



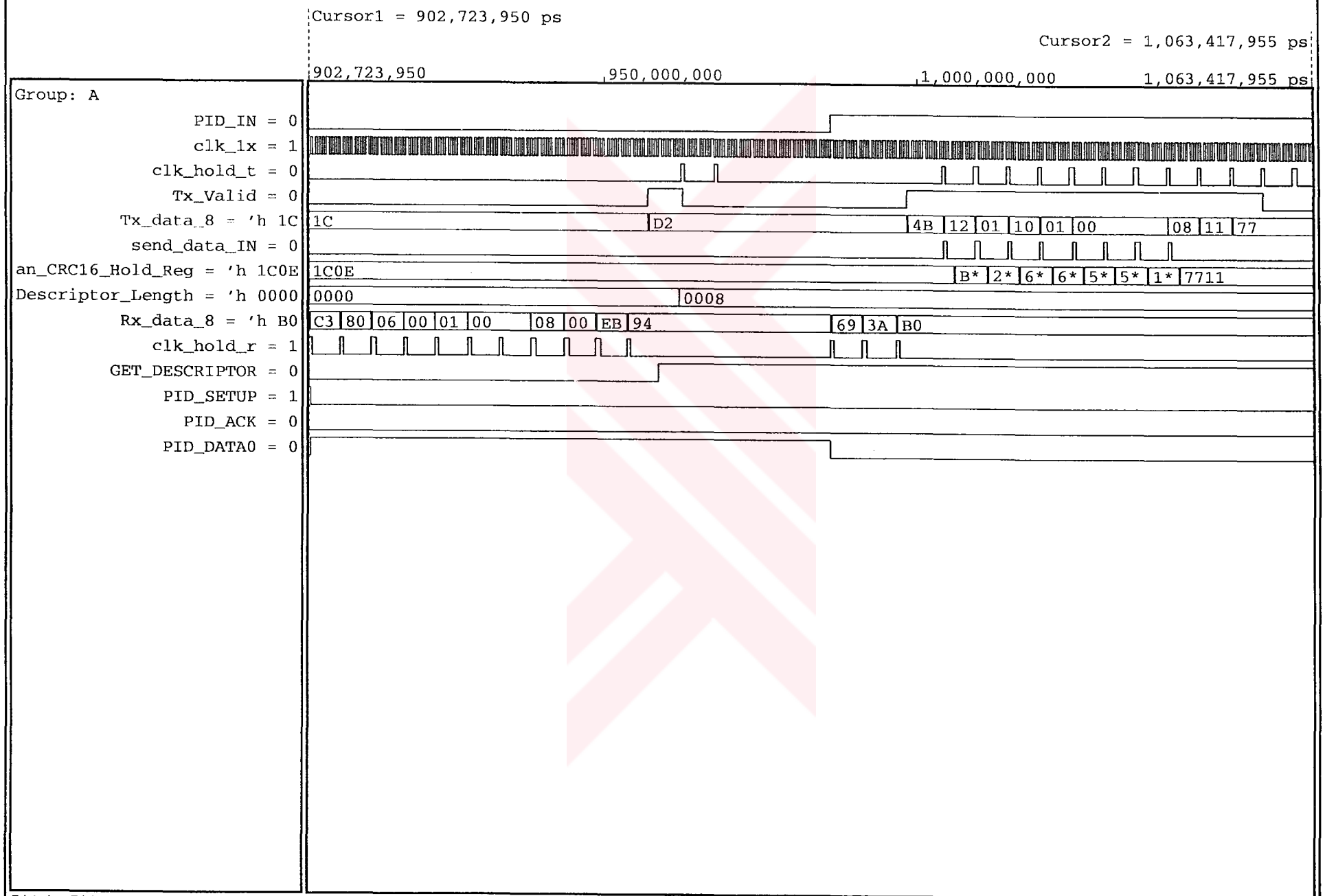
ITU ETA ASIC TASARIM MERKEZI

Veri Hattina Cikarilan Handshake Paketleri



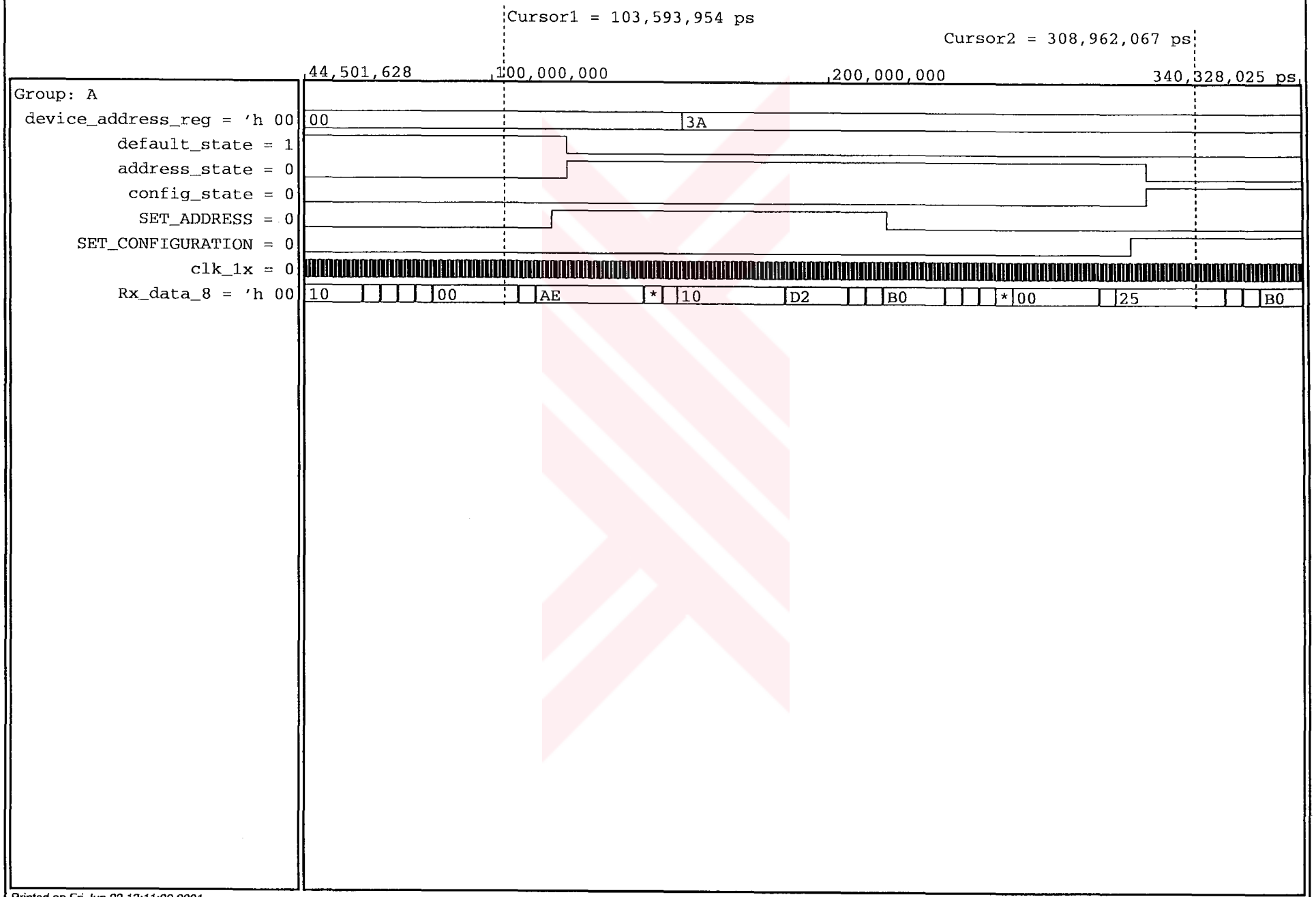
ITU ETA ASIC TASARIM MERKEZI

Cihaz Tanimlayici Verisinin Konfigurasyon Sirasinda Veri Hattina Cikarilmasi



ITU ETA ASIC TASARIM MERKEZI

SETUP Isleminde Kullanilan Standart Cihaz Istek Paketleri "SET_ADDRESS ve SET_CONFIGURATION"



ÖZGEÇMİŞ

Doğum tarihi	19.03.1978	
Doğum yeri	Eskişehir	
Lise	1992-1995	Eskişehir Fatih Fen Lisesi
Lisans	1995-1999	Yıldız Teknik Üniversitesi Elektrik-Elektronik Fak. Elektronik ve Haberleşme Mühendisliği Bölümü
Yüksek Lisans	1999-2001	Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Elektronik ve Haberleşme Müh. Anabilim Dalı, Elektronik Programı
İş Deneyimi	1999-	Y.T.Ü Elektronik ve Haberleşme Müh. Bölümü, Elektronik Anabilim Dalı, Araştırma Görevlisi