itüdergisi/***d***
mühendislik
Cilt:6, Sayı:1, 65-76
Şubat 2007

Dağınık kontrol sistemleri için bir ağ tasarımı

Gökay K. HURMALI*, Turgut B. KARYOT

İTÜ Fen Bilimleri Enstitüsü, Uçak Mühendisliği Programı, 34469, Ayazağa, İstanbul

Özet

Bu çalışmada, K-Bus olarak adlandırılan bir seri iletişim yolu çevresinde ölçeklenebilir, esnek ve modüler bir kontrol mimarisi önerilmiştir. RS–485 elektrik seviyesine donanım ilavesi yapılarak güvenli ve yazılım yükü çok olmayan veri çarpışma önleme sistemi kurulmuştur. Verilen donanım modeli üzerinde bir veri paketi iletim protokolü tanımlanmıştır. Yüksek gerçek zaman performanslı, göreli olarak az işlemci gücü gerektiren bir mimari anahtar tasarım istekleridir. Önerilen mimarinin düşük işlem güçlü mikroişlemcilerde kullanılabilmesine imkân tanımak için K-Bus ilgilenilmeyen veri paketlerini belirleyerek işlemciye iletimi kesme ve bir paket iletimi boyunca hatta erişimi engelleme kabiliyetine sahiptir. Bu şekilde işlemciye yüklenen protokol ile ilgili yazılım yükü azaltılmıştır. Önerilen mimarinin ayırt edici özelliklerinden birisi, tek çevrim içerisinde farklı ağ modüllerinde bulunan verilere ulaşabilme kabiliyetidir. Bu özellik hem modüllere yazarken hem de modüllerden okurken kullanılabilir. K-Bus'ın diğer ayırt edici özelliği ise modül ve veri adresi kullanmadan, dağınık gerçek zamanlı ağ üzerinden, verilere ulaşabilme kabiliyetidir. Bu ayırt edici özelliklerinin yanı sıra, K-Bus, CAN ve Profibus gibi diğer modern gerçek zamanlı veri yollarında bulunan kabiliyetlere de sahiptir. Birçok konuşucu modül aynı veri yolu üzerinde bulunabilir. Bir veri paketi yayın olarak birden çok modüle iletilebilir. Veri paketleri için öncelik verilebilir. Paket çarpışmaları donanım düzeyinde önlenir. Hatalı veya çarpışan paketler yazılım tarafından tekrar yollanır.

Anahtar Kelimeler: *Gerçek zamanlı ağlar, dağınık kontrol sistemleri, insansız hava araçları.*

*Yazışmaların yapılacağı yazar: Gökay K. HURMALI. hurmali@yahoo.com; Tel: 01(425) 703 50 79.
Bu makale, birinci yazar tarafından İTÜ Uçak ve Uzay Bilimleri Fakültesi'nde tamamlanmış olan "İnsansız hava araçları için modüler kontrol mimarisi tasarımı" adlı doktora tezinden hazırlanmıştır. Makale metni 16.11.2005 tarihinde dergiye ulaşmış, 20.06.2006 tarihinde basım kararı alınmıştır. Makale ile ilgili tartışmalar 31.10.2007 tarihine kadar dergiye gönderilmelidir.

Bus design for distributed control systems

Extended Abstract

In this study, modular, scalable and flexible control system architecture around a serial communication bus is proposed to fulfill control tasks. This bus is named K-Bus. One of the differentiating features of the proposed architecture is its capability to access different data on distributed modules in a single network transaction. Accessing multiple modules in a single transaction can be achieved both while data writing and data reading. Second differentiating feature of K-Bus is its capability to access predefined data without using module or data addresses. Network data collision avoidance is obtained by adding hardware components around RS–485 electrical layer to detect and avoid data collision. Package priorities are obtained with sender's identification number in the protocol, which is also used by the hardware layer to decide the winner in case of a collision. Two timer based locks are provided by the hardware layer blocks accessing to the bus during a transaction. One of the timers measures the time to end a whole transaction, while other timer keeps the time between different module or data cycles within the transaction. Modules replying to a high priority master will be able to produce their response in the same package cycle, without the need to wait for the next available slot. This allows high priority master modules to rapidly access to low priority modules within master module's priority. This feature enables predictable access times to resources in multi mastered networks. K-Bus also allows acknowledgement to be transferred in a single network transaction.

High real time performance and relatively less processor resource consumption are key design requirements. Signals between K-Bus hardware and microprocessor are designed to allow interrupt driven software for network interfacing. Hardware layer is designed to transfer minimal load to the processor. It is capable of blocking interrupts to the processor for data packages that are not targeted to processor's module. An analyze have been performed to show that, in the given example, using hardware provided method to block interrupts for a transaction that does not involve the module would reduce the protocol related CPU load from 18.75% to 3%. Proposed architecture enables K-Bus to be used in different applications with variety of modules.

K-Bus has following properties: different data from multiple modules can be accessed in a single transaction; multiple master modules can be present on the same data bus; hardware priority is given for data packages; data package collisions are detected at hardware level; erroneous or collided data packages are automatically resend by software; one data package can be broadcasted to more than one module; by the help of standard data concept, fundamental parameters can be reached without using module address; by the help of block data transfer, more than one data can be transferred in single K-Bus package and the overhead coming from package structure is reduced.

The paper is organized as follows: first the underlying electrical layer of K-Bus, RS-485, is briefly compared to other real-time network electrical layers. In the following section, logical components of K-Bus are described. These are given in Figures 1 and 2. An analysis is done in this section to demonstrate the CPU savings using K-Bus's interrupt disabling capability for packages that are not targeted to a module. In the next section a sample read package was shown in Figure 3. This sample read package demonstrates interrupt blocking capability and multiple data read capability. All K-bus packages have 9 fields: sender's identification, SID; receiver's identification, RID; command, CMD; package length, LEN; data fields DTA; optional acknowledge or not acknowledge, ACK/NAK; optional command data, CD; checksum CSM; and end of package, EOF. All K-Bus fields, sections except DTA, are of fixed length of 1 byte. Figure 3 shows all the sections except optional ACK/NAK. All K-Bus protocols start with sender identifier (SID) field. This field is used by the hardware layer to determine the priority of the sender. In case of data collision this field is used to decide the winner. Figure 3 also shows two hardware timers, HM and HM2, which are used to control transaction. In the later section a detailed electrical design of the given K-Bus interface is given and is shown in Figure 4. Last section discusses trade off of performance for real time capabilities in K-Bus design.

Keywords: *Real-time buses, distributed control systems, unmanned aerial vehicles.*

Giriş

Dağınık kontrol sistemleri için günümüzde çeşitli ağ standartları mevcuttur. Bu makalede, çeşitli ağ sistemlerinin gerçek zamanlı özellikleri ele alınarak ve bunlara tek çevrim içerisinde ağ üzerinden okuma yapabilme kabiliyeti eklenerek yeni bir ağ tasarımı sunulmuştur.

Ticari olarak mevcut gerçek zamanlı ağların çevresinde; çeşitli kontrol sistemleri, ölçüm ve tahrik cihazları için arabirimler, programlama ve analiz araçları bulunmaktadır. Bu tip ağların en çok kullanılanlarından biri Controller Area Network (CAN)'dır (Farsi vd., 1999; Zeltwanger, 1995). Bu tip bir ağ çevresinde ticari olarak mevcut birimleri kullanarak kontrol mimarisi kurulması ile denenmiş ürünleri kullanmak, programlama, hata bulma ve analiz araçlarının mevcudiyeti, geliştirme zamanının hızlı olması (Ford, 1998) gibi avantajlar elde etmek mümkündür.

Günümüzde kullanılan en yaygın ve modern veri yolları için hala uygulamaya yönelik veya ağın çeşitli özelliklerini geliştirme amaçlı eklemeler yapılmaktadır. Örneğin, CAN bus için otomatik modül bulma (Cena ve Valenzano, 2003) gerçek zamanlı haberleşme güvenilirliğini arttırma (Pinho ve Vasques, 2003), performansını arttırma (Cena ve Valenzano, 2000), ağ paylaşımının iyileştirilmesi (Zuberi ve Shin, 2000), bant genişliği paylaştırılması (Hong ve Kim, 2000) konularında çalışmalar son yıllar içerisinde gerçekleştirilmiştir.

Bu makalede, donanım seviyesi olarak RS–485 seri kanalına donanım ilavesi ile gerçekleştirilebilecek bir seri yol tasarımı sunulmuştur. Bu tasarım K-Bus olarak adlandırılmıştır. RS–485 seri kanalına donanım ilavesi yapılmasının nedeni güvenli ve yazılım yükü çok olmayan bir çarpışma önleme sistemi kurmak içindir. Bu makalede çarpışmadan kasıt, veri yolu üzerinde birden çok aygıtın aynı anda iletişim hattına çıkması ile oluşabilecek problemdir. Tasarlanan mimari için gerekli ek yazılım yükü azdır ve birçok işlemcide rahatlıkla uygulanabilir. Yüksek gerçek zaman performanslı, göreli olarak az işlemci gücü tüketen bir mimari, anahtar tasarım istekleridir.

CAN ile RS–485 arasında bir karşılaştırmada (Castro vd., 2002) gerçek zamanlı saat eşlemesi ve gerçek zamanlı nesne değişimi gibi özelliklerinden dolayı CAN tabanlı çözümün daha üstün olduğu sonucuna varılmıştır. RS-485 sadece donanım seviyesi tanımlarken, CAN donanımın üzerinde ki seviyeleri de tanımladığından dolayı bu karşılaştırma sadece donanım karşılaştırması olarak kullanılmalıdır.

Donanım tasarımı

K-Bus diferansiyel bir seri iletişim standardı olan RS–485 elektrik tanımlamalarını kullanmaktadır. K-Bus donanım modeli tanımlanırken ilk önce iki kısma bölünmüştür. RS-485 standardı ile uyumlu kısımlar ve K-Bus'ün özelliklerini elde etmek için gerekli mantık kısmı birbirinden ayrılmıştır. Şekil 1'de *KBCL* (K-Bus Circuitry Logic, K-Bus mantığının devre şeması) ile K-Bus'ın mantıksal kısmı gösterilmiştir. RS–485 standardını uygulamak üzere bu çizimde SN75LBC176 entegre devresi kullanılmıştır. Çizimde kullanılan sinyallere ait kısaltmaların özet açıklamaları Tablo 1'de verilmiştir. Şekil 1 bir işlemciye bağlanmak üzere tüm K-Bus sinyallerini göstermektedir. Bu çizimde *A* ve *B* sinyalleri K-Bus bağlantılarıdır.

Önerilen K-Bus yapısının temel sinyallerinin anlamları aşağıda açıklanmaktadır.

BPI sinyali *KBC* ile etkileşen işlemci tarafından üretilmektedir. Bu sinyal aktif olduktan sonra, *HM2* aktif olana kadar *KBC* devresi *RxD* çıkışını üretmeyecektir. İşlemci, gelen bir K-Bus paketinin *RID* (hedef kimliği) alanını aldıktan sonra, bu paketle ilgisi olmadığına karar verirse bu sinyali üreterek bu paket içerisindeki takip eden baytların alınmasını engelleyebilir. Bu şekilde işlemci kendisi ile ilgili olmayan bir pakette sadece iki veya üç bayt tarafından kesmeye uğramış olur: Birinci bayt SID, ikinci bayt RID, üçüncü bayt ise CMD'dir. İşlemci RID iletiminden sonra CMD başlamadan ilgilenmediği paketi belirleyip BPI sinyalini aktif hale getirebilirse CMD tarafından kesmeye uğramayacaktır. RID

bir FPGA veya benzeri bir mantık devresi ile süzgeçten geçirilerek BPI sinyali oluşturulduğunda bu sinyal CMD'den önce aktif hale getirilerek işlemcinin sadece ilk iki bayt tarafından kesintiye uğraması elde edilebilinir. Ayrıca işlemci K-Bus'a veri transfer ederken de BPI sinyalini üreterek kendi transfer ettiği verilerin kendisi tarafından okunmasını engelleyebilir. Ancak işlemciler veri transfer ederken paketin ilk baytı (*SID*) sırasında bu sinyali üretmemelidirler. Eğer çarpışma oluşursa ve işlemci kaybederse, kazanan modüle ait *SID* okunmuş olmalıdır.

HM sinyali *KBCL* tarafından üretilmektedir. Bir K-Bus paketinin *RID* alanında kendisine hitap edildiğine karar veren bir işlemci *HM* sinyali aktif olduktan sonra bu paketin *SID* alanında kimliği belirtilen modüle cevap vermek üzere K-Bus'a paket transferine başlayabilir. *HM* sinyali K-Bus üzerinde belli bir süre hareket olmadıktan sonra üretilir. Bu sinyal sayesinde bir modülden veri okumak isteyen bir diğer modül, araya başka K-Bus paketi girmeden veri okuyabileceğini garantileyebilir.

HM2 sinyali de *KBCL* tarafından *HM* sinyaline benzer şekilde üretilmektedir. *HM2* sinyalinin gecikmesi *HM* sinyalinin gecikmesinden daha fazladır. Bir önceki veri paketine cevap verme dışında tüm veri transferlerinden önce *HM2* sinyalinin aktif olduğu işlemci tarafından kontrol edilmelidir. Aksi takdirde K-Bus üzerinde bir veri transferinin ortasında yeni bir transfer başlatılıp transfer edilen paketin bozulmasına sebep olunabilir.

CP sinyali *KBCL* tarafından çarpışma oluştuğunda ve kaybedildiğinde üretilir. Bu sinyal, işlemci *E* sinyalini pasif hale getirene kadar tutulur. *KBCL* devresi *CP* sinyalini üretirken RS-485 entegre devresini de kapar ve K-Bus'a veri iletimini engeller. Çarpışma sinyali alan işlemci *E* sinyalini pasif hale getirip, *HM2* sinyalini beklemeli ve veri transferini tekrar denemelidir.

PB sinyali *KBCL* tarafından *BPI* sinyali alındığında aktif hale getirilmekte ve *HM2* sinyali aktif hale geldiğinde pasif hale geçmektedir. Bu sinyalin aktif olması şu anda K-Bus üzerinde bir transfer olduğunu ama bu transferin işlemciye iletilmediğini göstermektedir.

RxD sinyali, K-Bus donanımı aracılığı ile iletişimde olan donanımın işlemcisi tarafından değerlendirilmesi gereken K-Bus verisidir. Bu sinyal K-Bus üzerindeki çarpışmalar sırasında oluşan kısa süreli yanlış durumlardan arındırılmıştır ve *PB* aktif olduğu sürece pasiftir.

Tablo 1. KBC kısaltmalarının özet açıklamaları

Kısaltma	**Açıklama**
BPI	Bu paketle ilgilenilmiyor
HM	Hat müsait
HM2	Hat müsait 2
CP	Çarpışma
PB	Paket bekleniyor
RxD	Received Data
RxD0	Received Data 0
TxD	Transmitted Data
E	Enable
E2	Enable 2
A	K-Bus verisi
B	K-Bus verisinin değili
A1	KBCL'ye giren A
A2	KBCL'den çıkan A
VCC	Besleme gerilimi
VSS	Referans gerilimi

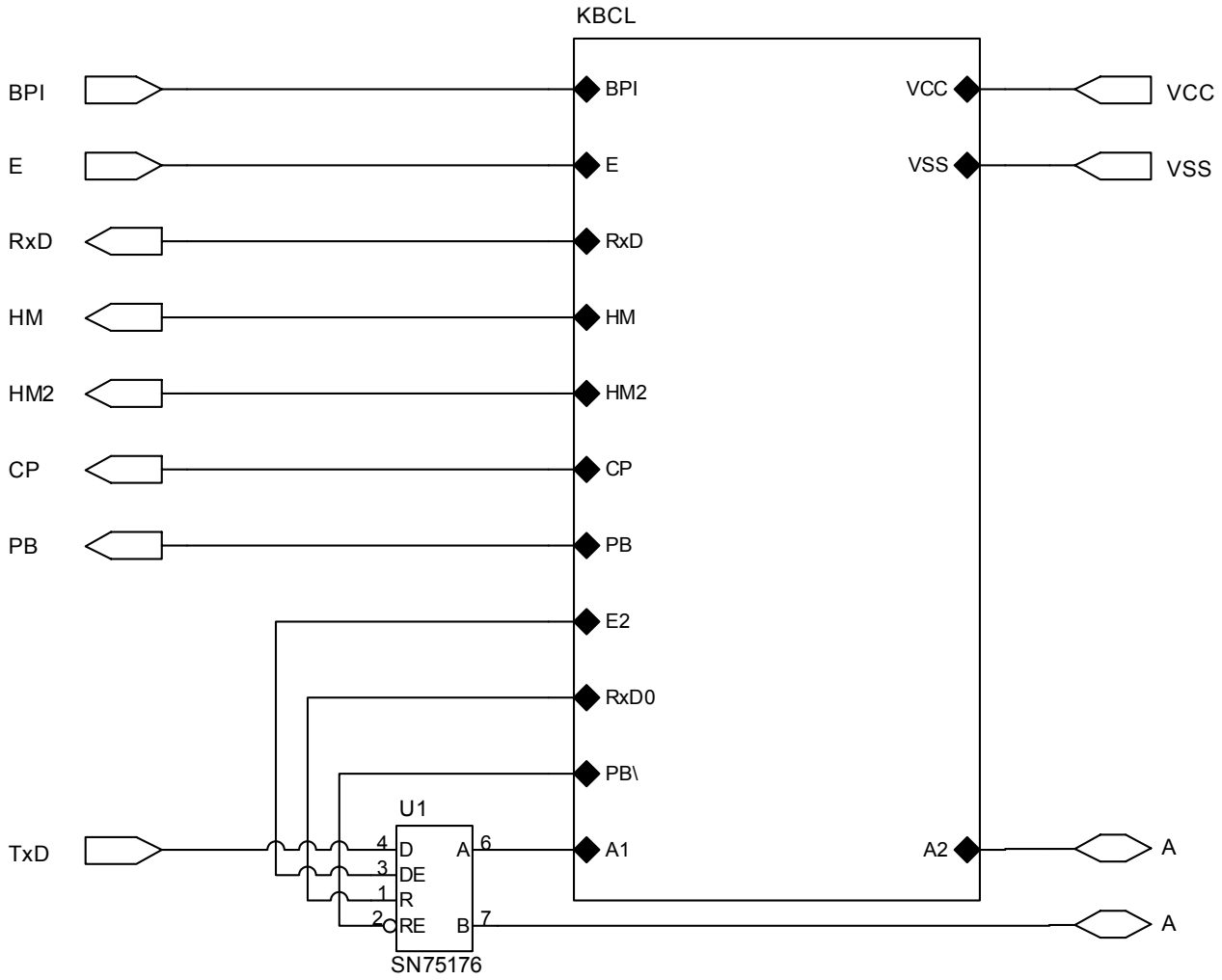
TxD sinyali, K-Bus aracılığı ile iletişimde olan donanımın işlemcisi tarafında üretilen K-Bus verisidir. *E* sinyali aktif ve *CP* sinyali pasif iken bu veri *KBC* tarafından K-Bus'a transfer edilir.

E sinyali, işlemci K-Bus'a veri transferine başlamadan önce üretilmelidir. İşlemci bu sinyali üretmeden önce *HM* veya *HM2* sinyallerini kontrol etmelidir. İşlemci K-Bus paketi biter bitmez, *HM* sinyali aktif olmadan önce, *E* sinyalini pasif hale getirmelidir.

A sinyali *KBC* tarafından K-Bus'a sürülen veya K-Bus'dan okunan veridir.

B sinyali *A* sinyalinin değilidir.

KBC'nin güç bağlantısı *VCC* ile gösterilmiştir



Şekil 1. K-BUS devresinin mantıksal şeması

VSS referans gerilimidir. Bu çalışmada tüm gerilim seviyeleri *VSS* sinyaline göre verilmiştir. *VCC* gerilimi ve toleransları (1)'de verilmiştir.

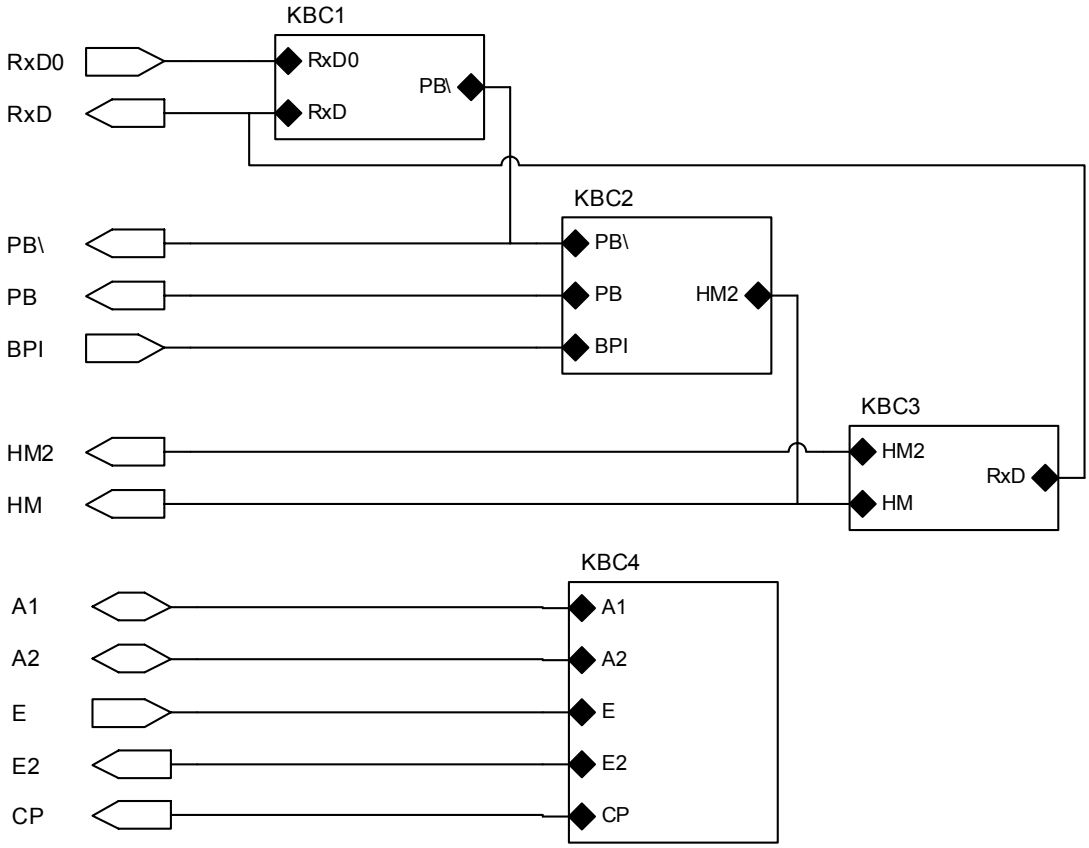
$$VCC \equiv 5\,V \pm 0.25\,V \qquad (1)$$

RxD0, E2, A1 ve *A2* sinyalleri *KBC* içerisinde *KBCL* ile hat sürücü arasında kullanılmaktadır.

KBCL devresi, açıklanmasını kolaylaştırmak için dört bloğa ayrılmıştır. Bu blokların aralarındaki ilişkiler Şekil 2'de gösterilmiştir. Şekil 2'de besleme gerilimleri çizilmemiştir.

KBC1 bloğu *RxD0* ve $\overline{PB}$ sinyallerini giriş olarak almaktadır. *RxD0* girişi K-Bus üzerinde bulunan veri, $\overline{PB}$ sinyali ise paket bekleniyor sinyalinin değilidir. $\overline{PB}$ sinyali aktif değil iken (1 mantık seviyesi) *RxD* çıkışı *RxD0* girişine eşit olmaktadır. $\overline{PB}$ sinyali aktif iken ise, *RxD* çıkışı 0 mantık seviyesinde kalmaktadır. Bu blok, paket bekleniyor sinyali aktif iken işlemciye K-Bus sinyallerinin iletilmesini engellemektedir. *KBC1* bloğu *RxD0* girişini süzmektedir. K-Bus üzerinde çarpışmalar sırasında veya başka nedenlerle oluşabilecek *100 ns*'ye kadar olan yanlış verilere izin verilmektedir. *KBC1* bloğu, *100 ns*'den küçük sinyalleri süzerek işlemciye yanlış veri iletimini engellemektedir.

KBC2 bloğu *BPI* ve *HM2* sinyallerini giriş olarak kabul etmekte ve *PB* ve $\overline{PB}$ sinyallerini ise çıkış olarak üretmektedir. *BPI* sinyalinin aktif hale geçmesi ile *PB* sinyali aktifleştirilmekte ve *HM2* sinyali aktif hale geçince de *PB* sinyali pasifleştirilmektedir. $\overline{PB}$ sinyali *PB* sinyalinin



Şekil 2. KBCL blokları arasındaki ilişkiler

değili olarak üretilmektedir. Bir K-Bus paketi alan modül, bu paketin ikinci baytı olan *RID* alanını aldığında, paket ile ilgilenip ilgilenmediğine karar verebilir. Modül paketle ilgilenmediğine karar verirse, *BPI* sinyali üreterek bu paketin kalan baytlarının, *KBC* tarafından işlemciye iletilmesini durdurur ve işlemci üzerinde gereksiz kesmeler oluşması engellenmiş olur. Düşük performanslı işlemciler kullanılarak basit giriş çıkış işlemlerini gerçekleştiren K-Bus modülleri tasarlandığında, *BPI* ve *PB* sinyalleri çok önem kazanır. Bu sinyaller modül üzerinde her K-Bus baytında kesme oluşmasını engeller. *BPI* ve *PB* sinyalleri kullanılmadığı durumda, sadece K-Bus aktivitesini gözlemlemek için, önemli bir işlemci gücü gerekecektir. Bu sinyaller, çok düşük performanslı işlemcilerle K-Bus modüllerinin gerçekleştirilebilmesine imkân tanır.

KBC3 bloğu *HM* ve *HM2* sinyallerini üretmektedir. *RxD* sinyalinde belli bir süre hareket olmayınca ilk önce *HM* sinyali daha sonra *HM2* sinyali üretilmektedir. K-Bus frekansı $F \equiv 500kHz$ olarak tanımlanmaktadır. K-Bus üzerinde RS-485 standardında belirtilen elektriksel büyüklük seviyeleri kullanıldığından dolayı, hat uzunluğuna bağlı olarak daha yüksek iletişim hızları mümkündür.

Verilen K-Bus iletişim hızında bir bitin iletim süresi

$$t \equiv 1/F = 1/500000 = 2\ \mu s \qquad (2)$$

olarak bulunur. K-Bus üzerinde her bayt 8 veri biti, 1 başlangıç biti ve 1 dur biti olmak üzere toplam 10 bitten oluşmaktadır. Bu değer $N_{Kb} = 10$ olarak gösterilsin.

Bu durumda K-Bus üzerinde bir bayt süresi

$$T = 10\,t = 20\ \mu s \qquad (3)$$

olarak elde edilir.

HM gecikme süresi $T_{HM} \equiv 2T = 40\ \mu s$ olarak tanımlanmıştır. Bu süre bir bayt süresinin iki katı olarak seçilmiştir. K-Bus üzerinde veri transferi gerçekleşirken, bir bayt içerisinde bulunan tüm bitler aynı olsa bile, başlangıç ve dur bitlerinden dolayı en az her *T* süresinde bir kez *RxD* sinyalinde değişim oluşacaktır. *HM* sinyalini kontrol eden işlemci K-Bus üzerinde şu anda iletilmekte olan paketle ilgili bir modül içerisinde bulunmaktadır. Bu modül yazılım olarak iletilmekte olan paketin sonunun geldiğini algılayacak ve daha sonra *HM* sinyalini kontrol ederek veri iletimine başlayabilecektir.

HM2 gecikme süresi $T_{HM2} \equiv 4T = 80\ \mu s$ olarak tanımlanmıştır.

K-Bus üzerinden yeni bir paket iletimi başlatmak isteyen modüller *HM2* sinyalinin aktif olmasını beklemektedirler. Modüller ilgili olmadıkları paketleri, paket iletiminin ikinci baytında belirleyip paketin kalan kısmını yazılım olarak analiz etmeyebilirler. Yeni iletim başlatmak için bekleyen modül sadece *HM2* sinyalini kontrol edecektir. Bu sebeple K-Bus'a veri transfer eden modüller, veri paketi içerisinde ardışık baytlar arasında en fazla 3 *T* gecikme olacağını garanti etmek zorundadırlar.

KBC4 bloğu diğer üç bloktan bağımsızdır. Bu blok K-Bus üzerinde oluşabilecek çarpışmaları algılayarak veri yolu sürücü entegresine giden *E2* sinyalini pasif hale getirmekten sorumludur. *KBC4* bloğu *E* giriş aktif hale geldiğinde *E2* sinyalini aktif hale getirmektedir. *KBC4* bloğu içerisinde *A1* ve *A2* girişleri 1Ω değerinde bir direnç ile birbirlerine bağlıdırlar. *A1* hattından *A2* hattına doğru pozitif olarak 100 mA veya üzeri bir akım, K-Bus üzerinde bir çarpışma olduğu ve bu modülün çarpışmayı kaybettiği anlamına gelmektedir. Bu durumda *KBC4* bloğu en geç 35 ns içerisinde *E2* çıkışını pasif hale getirmek zorundadır. *E* girişi aktif halde kalmaya devam etse bile *E2* çıkışı pasif halini koruyacaktır. *E2* çıkışını tekrar aktif hale geçmesi için *E* girişinin ilk önce pasif duruma geçmesi gerekmektedir.

Çarpışma durumunda kaybeden modül, akım yönü *A1*'den *A2*'ye doğru olan modül olduğundan dolayı, düşük modül adresleri daha önceliklidir.

Şekil 4'te kullanılan RS-485 hat sürücü entegre devresi, çıkışını kapatma sinyalini aldıktan sonra, en kötü durumda 65 ns içerisinde K-Bus çıkışlarını yüksek empedans durumuna geçirebilmektedir. *KBC4* bloğunun kısa devre oluşması ile RS-485 hat sürücü entegre devresine kapatma sinyalini üretmesi arasında, en fazla 35 ns gecikme olması gerekmektedir. Bu durumda en uzun kısa devre süresi bu iki sürenin toplamı olan $T_0 = 100\ ns$ olarak elde edilmektedir. Bu kısa devre süresince K-Bus üzerinde bulunan verinin hatalı olma ihtimali vardır. *KBC1* bloğunda bulunan süzgeç T_0'dan daha küçük süreli sinyalleri süzmektedir.

KBCL için bloklar arasındaki şematik ilişki Şekil 4'te referans bir tasarım ile gösterilmiştir.

PB sinyalinin gerekliliğini göstermek için örnek bir işlemci tanımı ve K-Bus yoğunluğu verilmiştir ve işlemcinin K-Bus ile ilgili işlemlerinin toplam işlemci kapasitesine oranı hesaplanmıştır.

K-Bus'ın ortalama $R_N = 0.3 = \%30$ doluluk oranı ile çalıştığı ve K-Bus üzerinde ortalama paket boyunun $N_{PL} = 30\ bayt$ olduğu kabul edilmiştir. Modül üzerinde bulunan işlemcinin $\Phi_{CPU} = 16\ MHz$ saat hızında çalıştığı, her seri ara birim kesmesinin $N_{CPU} = 100$ makine komutu ile işletildiği ve bu makine komutlarının ortalama $N_{\Phi N} = 2$ saat çevrimi sürdüğü kabul edilmiştir. Ayrıca işlemci üzerinde bulunan seri arabirim devresinin 1 bayt alım tampon bölgesi olduğu düşünülmektedir. Seri iletişim biriminde alıma ait tampon bölge 1 bayt olduğundan dolayı, K-Bus'dan gelen her bayt işlemcide bir kesme yaratacaktır. İşlemci için verilen değerler örnek tasarımda kullanılan Hitachi marka H8S serisi işlemci için gerçekçi değerlerdir.

Bu modüle saniyede $N_{PCPUN} = 50$ adet K-Bus paketi geldiği kabul edilmektedir. Öncelikle en zorlayıcı durum analizi gerçekleştirilecektir. K-Bus üzerinde en fazla saniyede

$$N_{KBMax} = F / N_{Kb} = 50000 \quad (4)$$

bayt transfer edilebilecektir. En uzun paket boyutu 261 bayt olarak sınırlıdır ve iki paket arasında en az *2 T* kadar gecikme olmak zorundadır. Bu paketler arası gecikme en kötü durum hesabını %1 mertebesinde etkilediğinden dolayı ihmal edilmiştir. K-Bus'ın tam olarak kullanıldığı durumda, işlemcinin K-Bus veri paketlerinden dolayı oluşan kesmelere harcayacağı saat çevrimi adedi

$$N_{clk} = 50000 \cdot 100 \cdot 2 = 10000000 \quad (5)$$

dur.

En zorlayıcı durumda örnekte verilen işlemcinin toplam kapasitesinin

$$CPU_{KM} = \frac{N_{clk}}{\Phi_{CPU}} = \frac{10000000}{16000000} = \%62.5 \quad (6)$$

kadarı harcanacaktır. Örnek olarak verilen orta performanslı işlemci ve zorlayıcı durumda K-Bus aktivitesini karşılayacak kapasiteye sahiptir. Verilen örnekte K-Bus doluluk oranı %30 olduğundan dolayı, *PB* sinyali kullanılmadığı durumda bu örnek için ortalama olarak işlemci kapasitesinin

$$CPU_{KN} = CPU_{KM} \times \frac{30}{100} = \%18.75 \quad (7)$$

kadarı kullanılacaktır.

Harcanan bu işlemci gücü önemli bir miktardır ve herhangi bir işe yaramadan heba edilmekte olan işlemci gücüdür.

PB sinyali kullanıldığı durumda, işlemci kendisine yöneltilmeyen paketlerin sadece ilk iki baytında kesmeye uğrayacaktır. Tüm K-Bus kapasitesi kullanıldığında saniyede paket sayısı en fazla

$$N_{PMax} = N_{KBMax} / N_{PL} = 50{,}000 / 30 \approx 1{,}667 \quad (8)$$

adettir. K-Bus doluluk oranı R_N olduğundan, ortalama saniyede paket adedi

$$N_{PN} = N_{PMax} \cdot R_N = 500 \quad (9)$$

olacaktır. İşlemciye saniyede ortalama N_{PCPUN} adet K-Bus paketi geldiği kabul edildiğine göre, işlemci her saniye içerisinde ortalama olarak (N_{PN}-N_{PCPUN}) adet paket için 2 kesmeye ve N_{PCPUN} paket için N_{PL} kesmeye maruz kalacaktır. İşlemcide ortalama olarak her saniye oluşacak kesme sayısı,

$$\begin{aligned} N_{ICPUN} &= (N_{PN} - N_{PCPUN}) \cdot 2 + N_{PCPUN} \cdot N_{PL} \\ &= 450 \cdot 2 + 50 \cdot 30 = 2{,}400 \end{aligned} \quad (10)$$

olarak hesaplanır. Eğer *PB* kullanılmasaydı işlemcinin servis vermesi gereken kesme adedi

$$N_{ICPUN2} = N_{PN} \cdot N_{PL} = 15000 \quad (11)$$

olacaktı. Verilen örnekte, *PB* sinyalinin kullanılması işlemcinin kesmelere servis vermekten doğan yükünü yaklaşık 6 kez azaltmaktadır. Verilen örnek için *PB* sinyali kullanıldığı durumda K-Bus analizinin işlemciye getirdiği ek yük toplam işlemci kapasitesinin

$$CPU_K = CPU_{KN} \cdot \frac{N_{ICPUN}}{N_{ICPUN2}} = \%3 \quad (12)$$

kadarını oluşturmaktadır.

Örnek okuma paketi

Bu bölümde tek çevrim ile K-Bus üzerinden bir okuma paketinin örneği verilmiştir. Şekil 3'te K-Bus üzerinde bulunan 3 modülün *RxD*, *TxD* ve *BPI* sinyalleri gösterilmiştir. Ayrıca tüm modüllerde ortak olarak bulunan *HM* ve *HM2* sinyalleri de verilmiştir. Bu örnek, K-Bus'ın en önemli ve özgün özelliklerinden biri olan tek veri paketi çevrimi içerisinde bir yollayıcının veriyi alabilmesi kabiliyetini göstermektedir. Bu özellik gerçek zamanlı dağınık denetim sistemleri için önemli bir kazanımdır. Bu örnekte, paketin toplam çevrim süresi yollayıcının *SID* baytını hatta sürmesi ile başlamakta ve *HM2* sinyalinin tekrar aktif hale geçmesi ile bitmektedir. Toplam paket çevrimi süresini hesaplamak için, arka arkaya gelen baytlar arasındaki bekleme süresi ve *HM* sinyalin aktif

hale geçmesi ile bu sinyali bekleyen modülün hatta veri sürmesi arasında geçen süre ihmal edilmiştir. Bu durumda toplam paket çevrim süresi:

$$\begin{aligned} T_{Paket} &= T_{SID} + T_{RID} + T_{CMD} + \\ &\quad T_{LEN} + T_{DTA0} + T_{DTA1} + \\ &\quad T_{HM} + T_{CD0} + T_{CD1} + + T_{HM} + \\ &\quad T_{CSM} + T_{EOF} + T_{HM2} \end{aligned} \quad (13)$$

şeklinde hesaplanmaktadır. Daha önce T_{HM} ve T_{HM2} süreleri, sırası ile $2T$ ve $4T$ olarak verilmişti. Diğer bayt sürelerinin hepsi de T olacaktır. Bu durumda, toplam paket çevrim süresi

$$\begin{aligned} T_{Paket} &= T + T + T + T + T + T \\ &\quad + 2T + T + T \\ &\quad + 2T + T + T + 4T \\ &= 18T \end{aligned} \quad (14)$$

şeklinde ortaya çıkmaktadır. (3)'i kullanarak

$$T_{Paket} = 18T = 18 \times 20\mu s = 360\mu s = 0.36ms \quad (15)$$

şeklinde hesaplanmaktadır.

Bu paketin transferi sırasında (13)'de bulunan iki adet T_{HM} ve bir adet T_{HM2} süresi boyunca seri iletişim yolu üzeride veri transferi yapılmamaktadır. Bu süreler dağınık sistemde ağı kilitlemek için ayrılmıştır. Bir paket içerisinde veri transferi için kullanılmayan süre

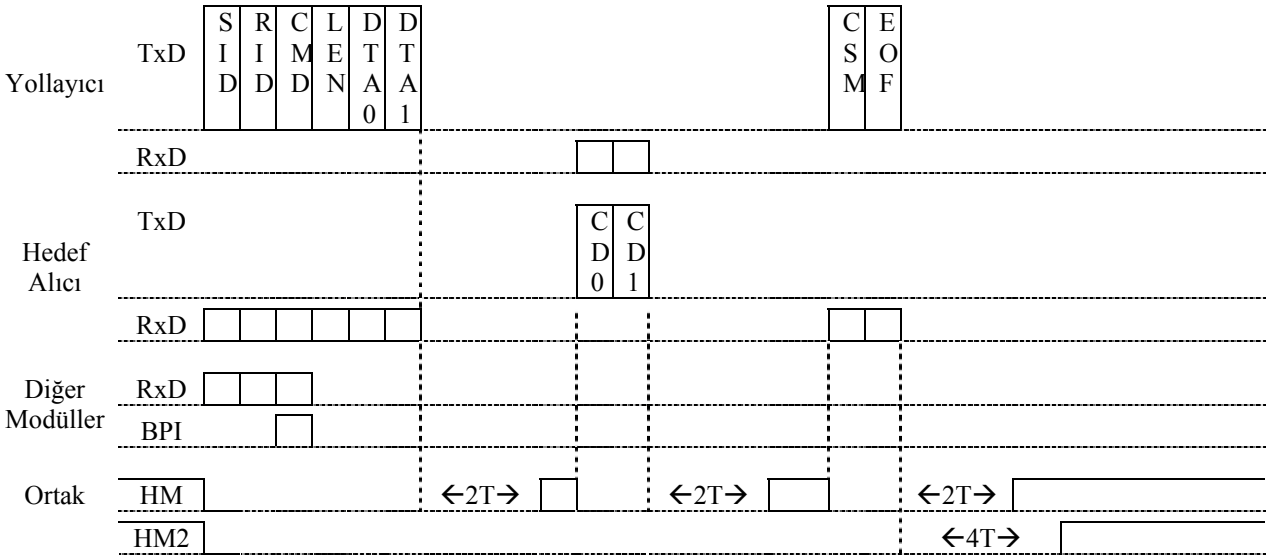
$$\begin{aligned} T_{Bos} &= T_{HM} + T_{HM} + T_{HM2} \\ &= 2T + 2T + 4T = 8T \\ &= 8 \cdot 20\mu s = 160\mu s = 0.16ms \end{aligned} \quad (16)$$

olarak hesaplanmaktadır. Örnekte verilen paket çevrimi içerisinde veri ağının bant genişliğinin kullanım oranı

$$\begin{aligned} R_{Kullanim} &= \frac{T_{Paket} - T_{Bos}}{T_{Paket}} \\ &= \frac{18T - 8T}{18T} = \frac{10T}{18T} \\ &\cong 0.56 = \%56 \end{aligned} \quad (17)$$

dır. T_{Bos} paket uzunluğuna bağlı olmadığından dolayı, daha uzun veri iletimi gerçekleştirilen paketlerde $R_{Kullanım}$ daha yüksek olacaktır.

Bu paketin transferi sırasında (13)'de bulunan iki adet T_{HM} ve bir adet T_{HM2} süresi boyunca seri iletişim yolu üzeride veri transferi yapılmamaktadır. Bu süreler dağınık sistemde ağı kilitlemek için ayrılmıştır. Bir paket içerisinde veri transferi için kullanılmayan süre



Şekil 3. K-Bus üzerinde örnek bir okuma paketi

$$
\begin{aligned}
T_{Bos} &= T_{HM} + T_{HM} + T_{HM2} \\
&= 2T + 2T + 4T = 8T \\
&= 8 \cdot 20\mu s = 160\mu s = 0.16ms
\end{aligned} \tag{18}
$$

olarak hesaplanmaktadır. Örnekte verilen paket çevrimi içerisinde veri ağının bant genişliğinin kullanım oranı

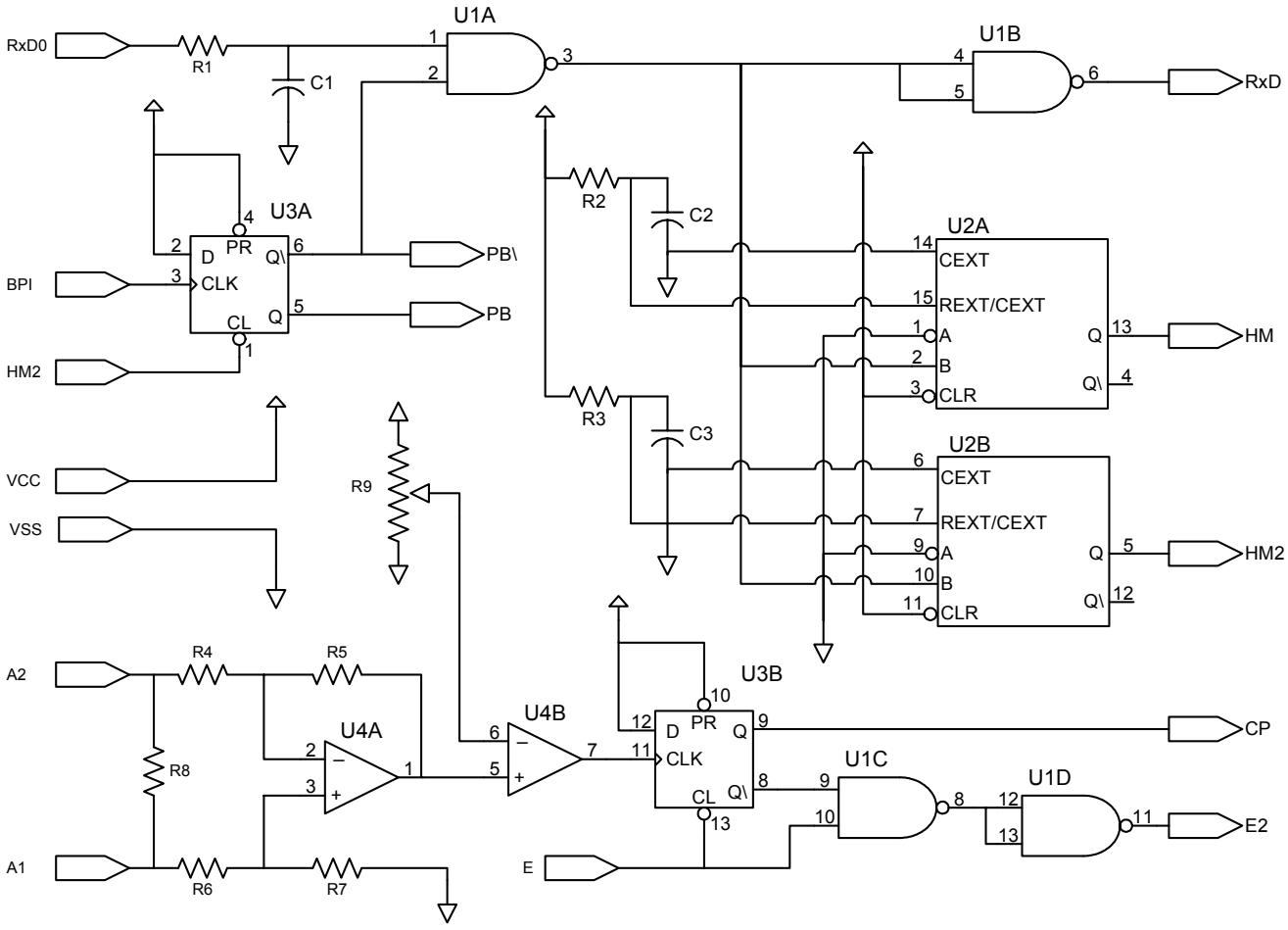
$$
\begin{aligned}
R_{Kullanim} &= \frac{T_{Paket} - T_{Bos}}{T_{Paket}} \\
&= \frac{18T - 8T}{18T} = \frac{10T}{18T} \\
&\cong 0.56 = \%56
\end{aligned} \tag{19}
$$

dır. T_{Bos} paket uzunluğuna bağlı olmadığından dolayı, daha uzun veri iletimi gerçekleştirilen paketlerde $R_{Kullanım}$ daha yüksek olacaktır.

Bahsedilen gerçek zaman özelliklerini sağlamak için, tek çevrimde okuma yapan paketlerin içerisinde veri iletimi için kullanılmayan bir süre eklenmiştir. (16)'da bu süre *8T* uzunluğunda olarak hesaplanmaktadır. Bu zaman ağın dağınık kilit denetimi için kullanılmaktadır. Bu zaman bir dezavantaj olarak görülse de, aynı paket içerisinde cevap iletilebilmesi bunu telafi etmektedir. Bu sayede, cevap için ayrı bir paket oluşturulmasına gerek kalmamakta ve cevap paketinin başlangıç ve bitişi için gereken zaman elimine edilmektedir. Mevcut birçok protokolde, elimine edilen bu süre, önerilen protokolde ki *8T* süresi ile karşılaştırılabilir uzunluktadır. Bu sayede toplam ağ kullanım oranı önemli ölçülerde zayıflatılmadan, önemli gerçek zaman kazanımları sağlanmıştır.

Referans K-Bus tasarımı

Şekil 1'de gösterilen ve Şekil 2'de ayrıntısı verilen KBCL bloğunun gerçekleştirilmesi için eleman seçimi ve şematik çizim bu bölümde yapılmıştır. Tasarıma ait şematik çizim Şekil 4'te gösterilmiştir.



Şekil 4. KBCL şematik tasarımı

Tablo 2. Referans KBCL tasarımında kullanılan malzemelerin değerleri

Sıra	Değer	Şematik ismi	Açıklama
1	1 KΩ	R1	Direnç.
2	4 KΩ	R2	Direnç.
3	20 KΩ	R3	Direnç.
4	40 KΩ	R4, R5, R6, R7	Direnç.
5	1 Ω	R8	Direnç.
6	50 KΩ	R9	Potansiyometre. Çıkışı 0.1 V olacak şekilde ayarlanacak
7	500 pF	C1	Kondansatör.
8	10 nF	C2, C3	Kondansatör.
9	74AHC00	U1A, B, C, D	Dört adet VE DEĞİL mantıksal kapısı içeren entegre devre.
10	74AHC123	U2A, B	İki adet gecikme devresi içeren entegre devre.
11	74AHC74	U3A, B	İki adet kapan içeren mantıksal entegre.
12	OPA2634	U4A, B	İki adet işlemsel yükselteç içeren entegre devre.

Tasarımda entegre devre olarak 74AHC serisi entegreler ve bir adet işlemsel kuvvetlendirici kullanılmıştır. Bu şematik çizimde kullanılan malzemeler Tablo 2'de verilmiştir.

Sonuçlar

Bu çalışmada gerçek zamanlı bir ağ önerilmiştir. Önerilen donanım modelinin belirleyici özellikleri önerilen sistemin tek çevrimde içerisinde okuma yapma kabiliyeti ve çarpışma önleme kabiliyeti olarak ortaya çıkmaktadır. Bu iki özellik aynı ağ içerisinde birden çok konuşucu modül olduğu durumda verilere erişim sırasında önceliklerin belirlenmesini sağlar. Bu sayede daha düşük öncelikli cevaplayıcı modüller yüksek öncelikli konuşucu modülün başlattığı çevrim içerisinde verilerini iletebilmektedirler. Bu özellik kontrol sistemlerinde cevap süresi tahmin edilebilir ağlar kurulması için kullanılabilir.

Bahsedilen gerçek zaman özelliklerini sağlamak için, tek çevrimde okuma yapan paketlerin içerisinde veri iletimi için kullanılmayan *8T* uzunluğunda bir zaman eklenmiştir. Bu zaman ağın dağınık kilit denetimi için kullanılmaktadır. Bu zaman bir dezavantaj olarak görülse de, aynı paket içerisinde cevap iletilebilmesi bunu telafi etmektedir. Bu sayede, cevap için ayrı bir paket oluşturulmasına gerek kalmamakta ve cevap paketinin başlangıç ve bitişi için gereken zaman elimine edilmektedir. Mevcut birçok protokolde, elimine edilen bu süre, önerilen protokolde ki *8T* süresi ile karşılaştırılabilir uzunluktadır. Bu sayede toplam ağ kullanım oranı önemli ölçülerde zayıflatılmadan, önemli gerçek zaman kazanımları sağlanmıştır.

Genişletilebilir ağ mimarisi ve gerçek zamanlı özellikleri, önerilen mimarinin endüstriyel ortamlarda ve insansız hava araçları gibi otonom cihazlarda kontrol sistemlerinin gerçekleştirilebilmesinde kullanılmasına imkan vermektedir. Verilen frekans değeri ve tek bir ağ üzerindeki modül sayısının sınırlılığı, önerilen model ile büyük ölçekli uygulamaların tüm ağ ihtiyaçlarının karşılanmasını engellemektedir. Ancak, büyük ölçekli sistemlerde gerçek zaman özelliklerinden feragat edilmesi beklenmelidir.

Kaynaklar

Castro, M., Sebastian, R., Yeves, F., Peire, J.; Urrutia, J., Quesada, J., (2002). Well-known serial buses for distributed control of backup power plants. RS-485 versus controller area network (CAN) solutions, *28th Annual Conference of the IEEE Industrial Electronics Society,* **3**, 2381 – 2386, Spain.

Cena, G., Valenzano, A., (2000). FastCan: A High-Performance Enhanced Can-Like Network, *IEEE Transactions on Industrial Electronics*, **47**, 951-963.

Cena, G., Valenzano, A., (2003). A Protocol for Automatic Node Discovery in CAN open Networks, *IEEE Transactions on Industrial Electronics*, **50**, 419-430.

Farsi, M., Ratcliff K. ve Barbosa, M., (1999). An Overview Of Controller Area Network, *Computing and Control Engineering Journal*, **10**, 113-120.

Ford, T., (1998). Actuation Systems Development, *Aircraft Engineering and Aerospace Technology,* **70**, 265-270.

Hong, S.H. ve Kim, W.H., (2000). Bandwidth Allocation Scheme in CAN Protocol, *IEE Proc.-Control Theory Appl.,* **147**, 37-44.

Pinho, L.M., Vasques, F., (2003). Reliable Real-Time Communication in CAN Networks, *IEEE Transactions on Computers,* **52**, 1594-1607.

Zeltwanger, H., (1995). An Inside Look at the Fundamentals of CAN, *Control Engineering,* **42**, 81-87.

Zuberi, K.M. ve Shin, K.G., (2000). Design and Implementation of Efficient Scheduling for Controller Area Network, *IEEE Transactions on Computers,* **49**, 182-188.