

Gömülü Çoklu İşlemcili Sistemlerde Yeniden Betimlenebilir Haberleşme Protokolleri

Salih Bayar, Arda Yurdakul

Boğaziçi Üniversitesi, Bilgisayar Mühendisliği Bölümü
P.K. 2 TR-34342 Bebek, İstanbul, TÜRKİYE
{salih.bayar, yurdakul}@boun.edu.tr

Özetçe

Gömülü çoklu işlemcili sistemlerde işlemciler arası haberleşme tıkanıklığını engellemek için son zamanlarda Yonga-üstü-Ağ(YüA) sıklıkla kullanılmaya başlanmıştır. Oysaki YüA'larda yönlendirme tablololu anahtarlar için gerekli olan ilave alan ihtiyacı, paketleme, yönlendirme ve anahtarlama için gerekli olan ilave gecikme zamanları YüA'ların haberleşme tıkanıklığını engellemek için en elverişli seçenek olmadığını göstermektedir. Bunun yerine, fazladan bir gecikmeye yol açmayan, bire bir doğrudan bağlantı bu tarz sistemlerde daima en iyi çözüm olmuştur. Örneğin [1] ve [2]'de kısmi yeniden betimlenebilir uçtan uca (U2U) haberleşme protokolleri ortaya atılmıştır. Bu çalışmada, [1] ve [2]'de anlatılan incelemeler bir adım daha ileriye götürülerek, kısmi yeniden betimleme işleminin devrenin çalışma esnasında yani dinamik bir şekilde, isteğe bağlı olarak, çoklu kez yapılması sağlanmıştır. Böylece, çalışma esnasında, devrenin geride kalan kısmına zarar vermeden, U2U haberleşme protokol yapılandırılması isteğe bağlı olarak değiştirilerek, YüA'a rakip bir yapı ortaya çıkarılmıştır. Bu yapı, APKD (Field Programmable Array Logic-Alan Programlanabilir Kapı Dizisi) tabanlı, otomotiv uygulamalarında kullanılan Çoklu Araç Takibi (ÇAT, Multi Target Tracking (MTT)) için optimize edilmiş gömülü çoklu işlemcili Yonga-üstü-Sistem (YüS) mimarisinde test edilmiştir [3].

1. Giriş

Gömülü çoklu işlemcili sistemler, tek işlemcili sistemlere göre, uygulamanın yapısına da bağlı olarak daha yüksek performans sağlayabilmektedir. Teorik olarak, YüS'deki paralel çalışan işlemci sayısı arttıkça performansında aynı oranda artması beklenmektedir. Ancak, işlemci sayısı arttıkça, var olan işlemciler arasındaki haberleşme protokolün yapısının da karmaşıklaşacağından dolayı, bu tarz sistemlerin tasarımı zorlaşmaktadır. İşte bu noktada, YüA yapısı, bu problemleri aşmak için literatür de önerilen çözümlerden [4]. Yakın ve diğer çözümler de esnek ve uyarlanabilir haberleşme yapılarıdır [5][6][7]. Lakin YüA'larda yönlendirme tablololu anahtarlar için gerekli olan ilave alan ihtiyacı, paketleme, yönlendirme ve anahtarlama için gerekli olan ilave gecikme zamanları YüA'ların haberleşme tıkanıklığını engellemek için en elverişli seçenek olmadığını göstermektedir. Bunun yerine, fazladan bir gecikmeye yol açmayan, bire bir doğrudan bağlantı bu tarz sistemlerde

daima en iyi çözüm olmuştur. Zaman içerisinde değişen farklı yapılandırmaların oluşturulması da APKD'nin, devrenin çalışma esnasında kısmi yeniden betimleme özelliği ile U2U haberleşme protokol yapılandırmalarının isteğe bağlı olarak değiştirilmesi ile sağlanmıştır.

İşlenen bilgi miktarı yoğun olan, gömülü çoklu işlemcili sistemlerde, aktörler arası haberleşme trafiği de oldukça fazladır. Haberleşme trafiğinin büyüklüğü de, uygulamanın yapısına ve aktör sayısına bağlı olarak değişiklik göstermektedir. Yine uygulamaya bağlı olarak, aktörler arası haberleşmenin önceden bilindiği durumlarda, bir sonraki yapılandırma, aktörler birbirleri ile haberleşme işlemine ara verip bilgi işlerken yapılabilir. Burada dikkat edilmesi gereken husus, dinamik kısmi yeniden betimleme (DKYB) işlemi ile yapılan yapılandırma değiştirme işlemi için gerekli sürenin, aktörlerin o anlık bilgi işleme sürelerinden daha kısa olması gerektiğidir. Anlatılan bu durum, aktörler arası haberleşme için mümkün olabilecek en iyi durumdur ve olası diğer bütün uygulamalardan (YüA, paylaşılmış veri yolu vb.) çok daha hızlıdır. Çünkü böyle bir durumda 2 ucun birbiri ile haberleşmesinde arada hiç bir fazladan devre olmadığından, fazladan herhangi bir gecikme olmayacaktır, var olan tek gecikme kablo gecikmesi olacaktır. Bu tarz bir gecikme, bütün değişken haberleşme yapıları için en ideal ve ulaşılabildiği mümkün olmayan bir durumdur. Bu yüzden adı geçen yöntemden en etkili şekilde faydalanılabilecek uygulamalar, bilgi işleminin yoğun olduğu ve DKYB süresinin aktörlerin o anlık bilgi işleme sürelerinden daha kısa olduğu uygulamalardır.

APKD'lerin kısmi yeniden betimlenmesi, APKD üzerinde devre çalışır durumda iken APKD'nin bir kısmının, diğer kısımların çalışmasını kısıtlamadan yapılandırmanın değiştirilebilirliği yeteneğidir [8]. Yani, genel olarak sistemde bir performans kaybı olmamıştır ya da tasarımın değişmeyen kısımların işlevselliğinde bir eksiklik olmamaktadır. Kısmi yeniden betimlenmenin avantajı kullanılarak, donanım kaynakları farklı uygulamalar arasında paylaşılabilir, böylece kaynak faydalanma oranı artırılarak devrenin toplam güç sarfıyatı düşürülebilmektedir [9]. Burada dikkat edilmesi gereken yeniden betimlenme için gereken gücün de göz ardı edilmemesi gerçeğidir. Kısmi yeniden betimlenmenin başka bir avantajı ise bu tarz sistemlerin herhangi bir zamanda güncellenebilmesi ve böylece devamlı bir şekilde donanım desteği sağlanabilmektedir [9].

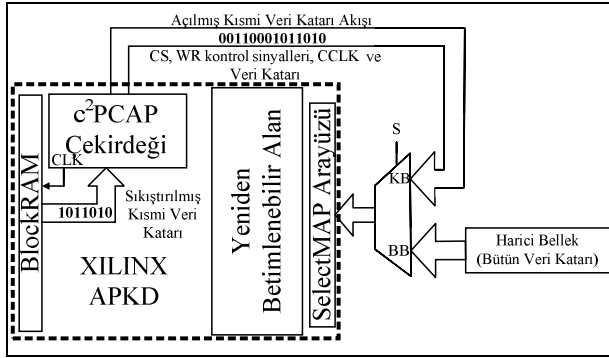
Bu çalışmada, dinamik kısmi yeniden betimlenmenin akışının kontrolü, daha önceden geliştirdiğimiz, cPCAP (compressed

Parallel Configuration Access Port) çekirdeğinin[10] geliştirilmiş hali olan c²PCAP (double compressed Parallel Configuration Access Port) çekirdeği[11] sayesinde sağlanmaktadır. c²PCAP çekirdeği önceden yaratılmış ve belirlenmiş olan sıkıştırılmış kısmi bit katarını çok hızlı Yonga-Üstü-Bellek (YüB) olan BlockRAM'de tutmaktadır. Böylece harici bir belleğe erişim ihtiyacı duyulmadan, c²PCAP çekirdeği her DKYB için YüB'de saklanan sıkıştırılmış kısmi bit katarını, zaman kaybına uğratmadan açarak, o tasarımda kullanılan APKD yapılandırma ara yüzü (SelectMAP ya da ICAP) aracılığıyla dinamik bir şekilde yapılandırma belleğine yazılmaktadır.

2. c²PCAP Mimarisi

Adı geçen c²PCAP çekirdeği, PCAP çekirdeğinin[12] geliştirilmiş bir şekli olup, önceden yaratılmış ve belirlenmiş olan, BlockRAM'de saklanan sıkıştırılmış kısmi bit katarlarının, isteğe bağlı olarak, zamanla kullanılan APKD yapılandırma ara yüzü aracılığıyla dinamik bir şekilde yapılandırma belleğine yazılmasını sağlayarak, DKYB için gerekli olan kontrol sinyallerini üretmektedir.

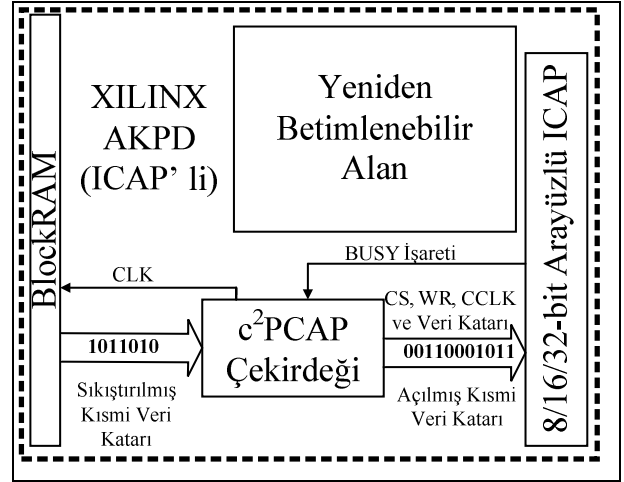
Şekil 1: Harici SelectMAP ara yüzü ile sistem donanımının mimarisi(KB: Kısmi Betimleme, BB: Bütün Betimleme)



XILINX APKD'leri c²PCAP çekirdeğinin kontrolü altında paralel olan harici SelectMAP APKD yapılandırma ara yüzü aracılığıyla, kendi kendini yeniden betimleyebilir. Burada, SelectMAP ara yüzü kullanılarak, kısmi veri katarı bu ara yüz tarafından kabul edilir ve kısmi yeniden betimleme işlemi aynı hedef olan APKD ile yapılır. Şekil 1'de görüldüğü gibi tek bir çip olan, kısmi olarak yeniden betimlenen APKD sadece edilgen değil aynı zamanda etkindir. SelectMAP ara yüzü sadece kısmi yeniden betimleme işlemi için ayrılmamıştır. Kısmi Betimlenmenin (KB) olmadığı zamanlarda, bu ara yüz aynı zamanda harici bellek ile APKD arasında Bütün Betimlenme(BB) amacı ile de kullanılabilir.

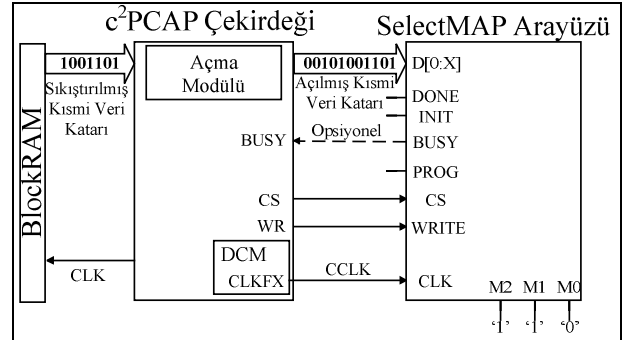
Benzer bir yapı İçsel Konfigürasyon Erişim Ara Yüzü'ne (İKEA, Internal Configuration Access Port(ICAP)) sahip olan (örneğin: Virtex-II, Spartan3A, Spartan-6 vb.) XILINX APKD'leri için kullanılmaktadır. Şekil 2'den görüleceği üzere, ICAP, SelectMAP benzeri olup, tamamen aynı görevi gören ancak yonga içinde bulunan bir APKD yapılandırma ara yüzüdür.

Şekil 2: Dâhili Yapılandırma Ara yüzü Üzerinden Sistem Donanım Yapısı



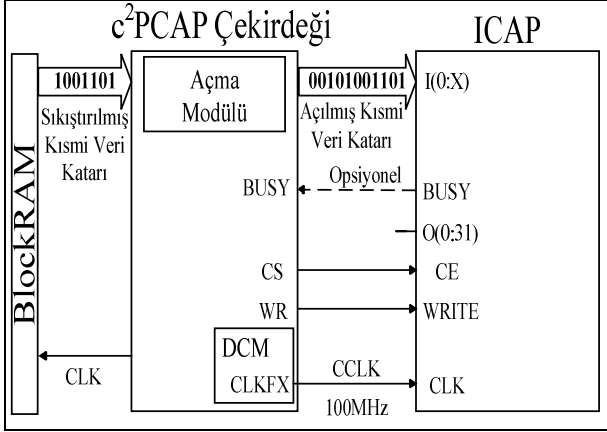
Burada dikkat edilmesi gereken ICAP'ın ara yüz genişliği (8/16/32-bit) yongadan yongaya farklılık gösterebilmektedir. Örneğin Virtex-II için 8-bit[13][14], Spartan-6 için 16-bit [15], Virtex-4 için 8/32-bit[16], Virtex-5 ve Virtex-6 için 8/16/32-bit[17][18] şeklindedir. Şekil 2'de görüldüğü gibi, yapılandırma ara yüzü için fazladan ne bir harici tele ne de harici bir elektriksel bağlantıya ihtiyaç vardır.

Şekil 3: c²PCAP Çekirdeği ve SelectMAP ara yüzü (X:7/15/31)



Şekil 3 ve Şekil 4'den anlaşılacağı üzere Yapılandırma Saat Frekansı'nı (YSF, Configuration Clock (CCLK)) üretmek için APKD'de mevcut bulunan Sayısal Saat Yönetici (SSY, Digital Clock Manager (DCM)) bloğu kullanılmaktadır. c²PCAP çekirdeği her bir saat döngüsünde yapılandırma ara yüzünün genişliğine bağlı olarak 1/2/4 Baytlık kısmi yeniden betimleme veri katarını BlockRAM'den okumaktadır. CS, WRITE ve CCLK sinyallerinin kontrolü altında, okunan bu veri kullanılan yapılandırma ara yüz tipine bağlı olarak SelectMAP ya da ICAP ara yüzüne gönderilir.

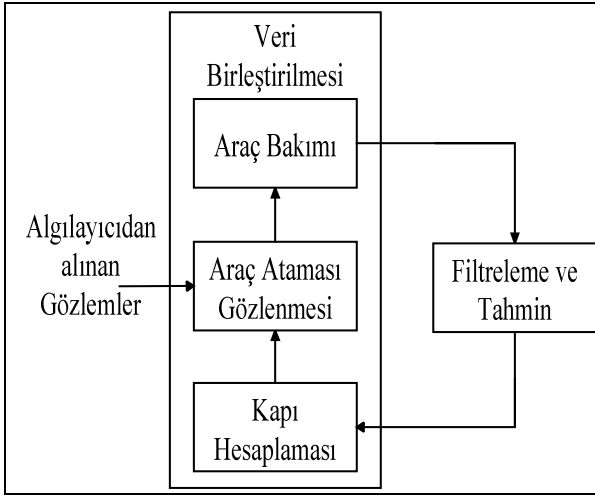
Şekil 4: c²PCAP Çekirdeği ve ICAP ara yüzü (X:7/15/31)



3. Örnek Olay İncelemesi: ÇAT

[3]'te APKD (Field Programmable Array Logic-Alan Programlanabilir Kapı Dizisi) tabanlı, otomotiv uygulamalarında kullanılan Çoklu Araç Takibi (ÇAT) için optimize edilmiş gömülü çoklu işlemcili Yonga-üstü-Sistem(YüS) mimarisi tasarlanmıştır. Şekil 5'te ÇAT sisteminin geliştirilmiş bir görünümü gösterilmektedir. Bu sistem Veri Birleştirilmesi ve Filtreleme & Tahmin olmak üzere 2 ana bloğa ayrılmıştır. Bu iki blok kapalı bir döngü içinde çalışmaktadır. Veri Birleştirilmesi bloğu kendi içinde 3 alt bloğa ayrılmıştır: Araç Bakımı, Araç Ataması Gözlenmesi ve Kapı Hesaplaması.

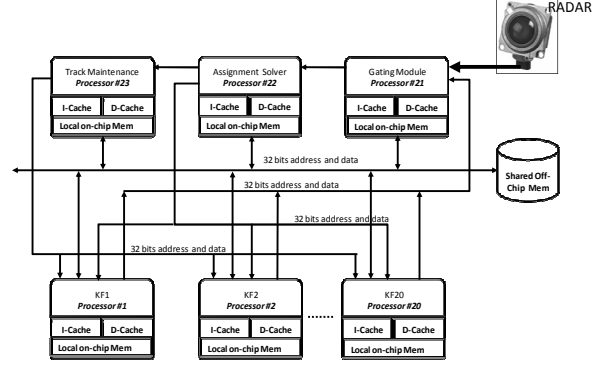
Şekil 5: ÇAT sisteminin basitleştirilmiş bir görünümü



Şekil 6'dan görüldüğü üzere bu yapıda toplam 23 adet Nios-II gömülü işlemci kullanılmış olup, işlemciler arası haberleşme değişmez zaman paylaşımı şeklindedir. Bu sistemde Filtreleme & Tahmin bloğu 20 adet Nios-II/s (Processor#1- Processor#20), Kapı Hesaplaması bloğu 1 adet Nios-II/f (Processor#21), Araç Ataması Gözlenmesi bloğu 1 adet Nios-II/s (Processor#22) ve Araç Bakımı bloğu 1 adet Nios-II/e (Processor#23) üzerinde gerçekleştirilmiştir.

Şekil 6'da ÇAT için optimize edilmiş gömülü çoklu işlemcili YüS mimarisindeki ilk kullanılan haberleşme yapısı da gösterilmektedir. Gömülü işlemciler arasındaki haberleşmeyi sağlayan bu yapı 3 adet paylaşımlı veri yolundan oluşmaktadır. Adı geçen bu örnek olay çalışması ALTERA APKD'leri üzerinde gerçekleştirilmiştir. Bizim çalışmamızda ise biz bu yapıyı XILINX APKD'leri üzerine taşıdık.

Şekil 6: İlk Gerçekleştirme: 3 paylaşımlı veri yolu



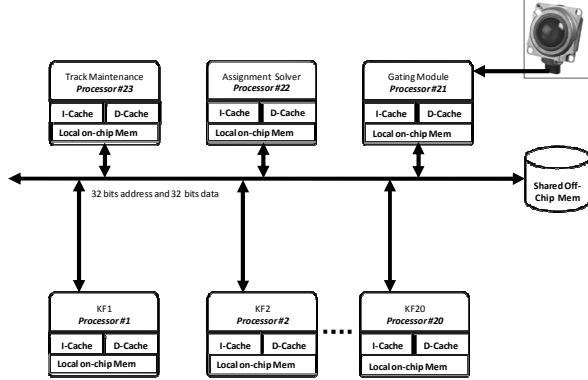
4. ÇAT'ın DKYB ile U2U Haberleşme Yapısının Gerçeklenmesi

Şekil 7-Şekil 11'de detaylı bir şekilde anlatıldığı üzere, sıralı, arka arkaya ve zamanla tekrarlı bir şekilde çalışan 5 farklı senaryo bulunmaktadır. Burada sadece 1 adet senaryo seçilmeyip, her bir senaryo zamanla, tekrarlı bir şekilde kısmi kendi kendine yeniden betimlenme ile gerçekleştirilmektedir. Uygulamanın yapısı gereği, işlemciler arası haberleşme yapısı zamanla Şekil 7'deki senaryodan başlayıp sırasıyla Şekil 11'deki senaryodaki gibi değişmektedir. Bizim burada gerçekleştirdiğimiz, işlemciler arası haberleşme yapısını senaryolarda belirtildiği şekilde sırası ile çalışma zamanında kısmi kendi kendine yeniden betimlenme ile değiştirmemizdir. Senaryolar arasındaki geçiş bütün betimlenme ile değil, çalışma zamanında kısmi kendi kendine yeniden betimlenme yöntemi ile gerçekleşmektedir. Çünkü var olan sistemde, çalışma zamanında hemen hemen bütün işlemciler etkin bir şekilde çalışmaktadır. Eğer kısmi yerine bütün yeniden betimlenme tercih edilseydi, sistemde performans düşüklüğü gerçekleşirdi. Bilindiği üzere yeniden betimlenebilir bir mimaride sabit, yani yeniden betimlenmeyen kısım ve bir de yeniden betimlenebilir kısım olmak üzere iki farklı kısım bulunmaktadır. Bizim oluşturduğumuz bu yapıda, 23 adet Nios-II gömülü işlemciler ve kontrol amaçlı kullanılan c²PCAP çekirdeği mimarininin sabit kısmında bulunurken, Nios-II gömülü işlemciler arasındaki bulunan farklı haberleşme yapıları yeniden betimlenebilir alanda durmaktadır. Çalışmanın detayları için ilgili makale[3] incelenebilir.

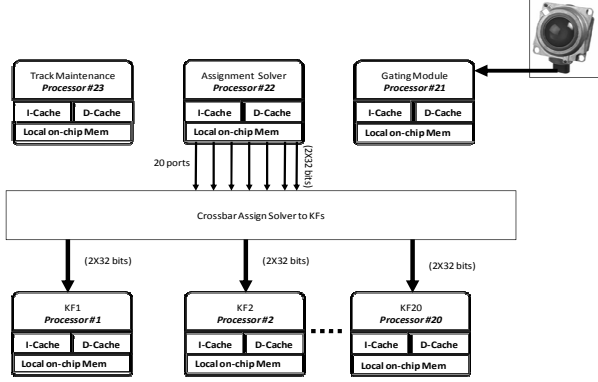
Bu çalışmada, ardışık 2 radar tarama zaman aralığı olan Darbe Yenileme Zamanı(DYZ, Pulse Repetition Time (PRT)) 25 milisaniyedir. Bu yüzden, işlemciler arası 5 farklı haberleşme yapısının değişimi için kullanılan toplam kısmi yeniden betimlenme süresi DYZ'den küçük olmalıdır. Buna ek olarak, 1 haberleşme yapısının değişimi için gerekli olacak

en uzun yeniden betimlenme süresi, herhangi bir işlemci için geçerli olan en kısa işlem zamanından daha kısa olmalıdır.

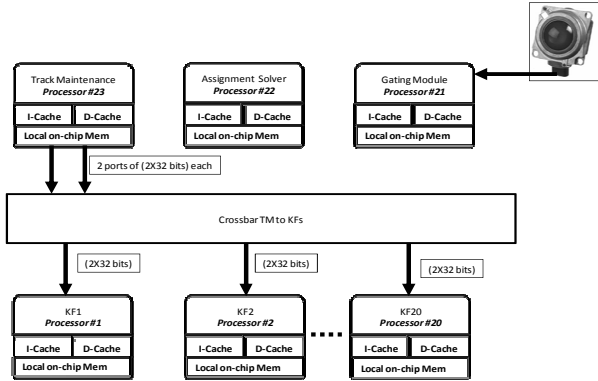
Şekil 7: [K1]: Veri ve kod erişimi için paylaşılmış veri yolu



Şekil 8:[K2]: Karşılıklı, birebir: Görev Çözücü'den KF Haberleşmesine



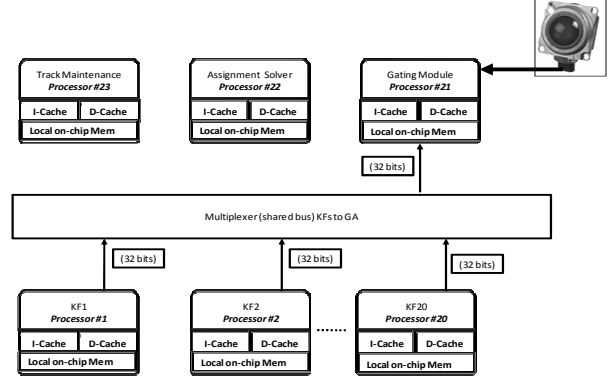
Şekil 9:[K3]: Karşılıklı, birebir: Araç Bakımı'ndan KF Haberleşmesine



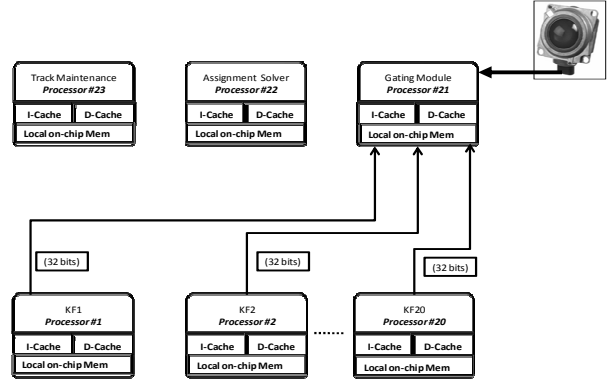
Hesap işleme zamanı, kısmi yeniden betimlenme süresinden daha uzun olduğundan dolayı ($T_{comp} \geq T_{reconf}$), hesap işleme zamanı ile kısmi yeniden betimlenme süresinin tamamen örtüşüğünü söylemek emniyetli olup, yeniden betimlenebilir

U2U haberleşme mimarisinin yapısının en elverişli şekilde kullanıldığı söylenebilir.

Şekil 10: [K4]: Çoğullayıcı: KF'den Sinyal Ayırıcıya



Şekil 11: [K5]: KF'den Sinyal Ayırıcı'ya Çoklu Alçak Frekans Veri Yolları



5. Deneysel Sonuçlar ve Değerlendirilmesi

Tablo 1: ÇAT Uygulamasındaki 5 farklı haberleşme senaryosu için kullanılan yeniden betimlenme süreleri

Kısmi Veri Katarı	Yeniden Betimlenme Süresi [ms]			
	ICAP 8-Bit		ICAP 32-Bit	
	50MHz	100MHz	75MHz	100MHz
K1	5.36	2.68	0.894	0.670
K2	5.36	2.68	0.894	0.670
K3	5.77	2.88	0.962	0.721
K4	5.47	2.73	0.911	0.683
K5	5.21	2.60	0.868	0.651
Total	27.17	13.57	4.529	3.395

Bu çalışmada, biz, Şekil 7-Şekil 11'de gösterilen çalışmanın haberleşme yapısını XILINX firmasının Virtex-4LX100 APKD'sinde gerçekleştirdik. Her bir farklı senaryo için yeniden betimlenme süreleri Tablo 1'de özetlenmiştir.

Tablo 1’den görüldüğü üzere en uzun süre örneğin K2’den K3’e geçişte 2,88 ms olup (100MHz, ICAP 8-Bit), en hızlı hesap işleme zamanından daha kısa süreli olması gerekir.[3] ‘te Tablo 3’ten anlaşılacağı üzere, en kısa süreli hesap işleme zamanı 8 ms olup Araç Bakım Bloğuna aittir. Bu vesileyle bakıldığında, Tablo 1’ de detaylı bir şekilde gösterildiği gibi bu koşulu (<8ms) bizim uyguladığımız herhangi bir frekans değeri ve herhangi bir ara yüz sağlamaktadır.

ICAP 8-bit, 50MHz seçeneğinde herhangi bir yeniden betimlenme süresi en kısa süreli hesap işlemekten daha kısa olmasına rağmen, toplam yeniden betimlenme süresi(27.17 ms) bu seçenek için DYZ’ den (25 ms) büyük olduğu için, bu seçeneğin (ICAP 8-bit 50 MHz) uygulanması imkânsız gözükmektedir. Oysaki ICAP 8-bit, 100MHz seçeneğinde hem toplam yeniden betimlenme süresi DYZ’ den daha küçük, hem de her bir kısmı yeniden betimlenme süresi en kısa hesap işleme süresinden daha az vakit almaktadır. Bunlara ek olarak, ICAP 32-bit modu takriben 15MHz den yüksek olmak koşuluyla herhangi bir frekans değeri için uygulanabilir durumdadır.

Bu deneyde her bir kısmı veri katarı takriben 260Kbyte civarındadır. Veri katarlarını sıkıştırarak, takriben %70 alan tasarrufu ile her bir veri katarı 79Kbyte’a inmiştir. 79Kbyte’lık bir veri takriben 40 adet yonga üstü BlockRAM bloğunu gerektirmektedir. Sonuç olarak, toplam kısmi veri paketlerinin (K1, K2, ...) miktarı 400Kbyte olup, 200/240 adet BlockRAM bloğu gerektirmektedir.

Uygulamanın ihtiyaçlarına göre (örneğin yüksek hız ya da kısıtlı miktarda YüB kullanım izni), bütün kısmi veri katarlarının saklanması YüB’ de olması yerine sadece 1 tanesinin YüB’ de, diğerlerinin harici bellekte saklanması ve YüB’ nin önbellek şeklinde kullanılmasıyla, YüB kullanımı azaltılabilir. Eğer önbellek yaklaşımı kullanılırsa takriben 40/240 BlockRAM bloğu (Toplamın %17 si) kullanılmış olur.

6. Sonuç

Bu çalışmada c²PCAP çekirdeği kullanılarak ICAP’ e sahip olan ya da olmayan bütün XILINX APKD’ lerin kendi kendine kısmi yeniden betimleme işlemini nasıl yaptığını, kısmi veri katarlarının nasıl YüB’ de tutulduğunu tartıştık. Bu yapı, APKD tabanlı, otomotiv uygulamalarında ÇAT için optimize edilmiş gömülü çoklu işlemcili Yonga-üstü-Sistem(YüS) mimarisinde test edilmiştir.

7. Kaynakça

1. *Partially Reconfigurable Point-to-Point Interconnects in Virtex-II Pro FPGAs*. **Jae Young Hur, Stephan Wong ve Stamatis Vassiliadis**. Mangaratiba, Brazil : s.n., March 27-29, 2007. Reconfigurable Computing: Architectures, Tools and Applications, Third International Workshop, ARC 2007.
2. *Partially Reconfigurable Point-to-Point FPGA Interconnects*. **Jae Young Hur, Stephan Wong ve Stamatis Vassiliadis**. 7, s.l. : International Journal of Electronics, Temmuz 2008, Cilt 95, s. 725 - 742.

3. *Trade-Off Exploration for Target Tracking Application in a Customized Multiprocessor Architecture*. **Jehangir Khan, Smail Niar, Mazen A. R. Saghir, Yassin El-Hillali ve Atika Rivenq-Menhaj**. 2009, EURASIP Journal on Embedded Systems.
4. *Networks on Chips: A New SoC Paradigm*. **L. Benini ve G. De Micheli**. Ocak 2002, IEEE Computers, s. 70-78.
5. *Dynamic Interconnection of Reconfigurable Modules on Reconfigurable Devices*. **Christophe Bobda ve Ali Ahmadinia**. 5, s.l. : IEEE Design & Test, September 2005, Cilt 22.
6. *Topology adaptive network-on-chip design and implementation*. **T. A. Bartic, J.-Y. Mignolet, V. Nollet, et al.** 4, 2005, IEE Proceedings: Computers and Digital Techniques, Cilt 152, s. 467-472.
7. *A study on communication issues for systems-on-chip*. **C. A. Zeferino, M. E. Kreutz, L. Carro ve A. A. Susin**. 2002, Symposium on Integrated Circuits and Systems Design Proceedings IEEE.
8. *Xilinx Development System Reference Guide 12.1i*. **Xilinx**. 2010.
9. *Benefits of partial reconfiguration*. **Cindy Kao**. s.l. : Xilinx, 2005, Xcell Journal, s. 65-67.
10. *Self-Reconfiguration on Spartan-III FPGAs with Compressed Partial Bitstreams via a Parallel Configuration Access Port (cPCAP) Core 2*. **Salih Bayar ve Arda Yurdakul**. s.l. : PRIME 2008 - 4th Conference on Ph.D. Research in Microelectronics and Electronics.
11. *An Efficient Self-Reconfiguration Core for Runtime Reconfigurable FPGA Interconnects*. **Salih Bayar ve Arda Yurdakul**. s.l. : IEEE Transactions on Very Large Scale Integration Systems, TVLSI (under revision), 2010.
12. *Dynamic Partial Self-Reconfiguration on Spartan-III FPGAs via a Parallel Configuration Access Port (PCAP)*. **Salih Bayar ve Arda Yurdakul**. Goteborg, Sweden : Proceedings of HiPEAC Workshop on Reconfigurable Computing, January 27, 2008.
13. *Virtex-II Pro and Virtex-II Pro X FPGA User Guide*. **Xilinx**. March 28 2007, UG012, (v4.1) edition.
14. *Virtex-II Platform FPGA User Guide*. **Xilinx**. March 28 2007, UG002 (v2.1) edition.
15. *Spartan-6 FPGA Configuration*. **Xilinx**. : s.n., February 22 2010, UG380, (v2.1) edition.
16. *Virtex-4 Configuration Guide*. **Xilinx**. June 9 2009, UG071, (v1.11) edition.
17. *Virtex-5 FPGA Configuration User Guide*. **Xilinx**. August 14 2009, UG191, (v3.8) edition.
18. *Virtex-6 FPGA Configuration*. **Xilinx**. January 18 2010, UG360, (v3.0) edition.