Bulut Bilişim için Donanım Hızlandırıcılar: Özellikler ve Gerçekleştirim
Hardware Accelerators for Cloud Computing: Features and Implementation

Anıl Tırlıoğlu*†, Ömer Bayram Demir†, Alper Yazar*†, Ece Güran Schmidt†

† Elektrik Elektronik Mühendisliği Bölümü, ODTÜ, Ankara, Türkiye
{anil.tirlioglu, demir.omer, alper.yazar, eguran}@metu.edu.tr
* ASELSAN A.Ş., Ankara, Türkiye
{atirlioglu, ayazar}@aselsan.com.tr

***Özetçe* —Bu bildiride, bulut bilişim sistemlerinde servis olarak sunulabilecek FPGA üzerinde gerçekleştirilen donanım hızlandırıcı (FDH) uygulamaları ele alınmaktadır. FDH uygulamalarının kullandıkları donanım kaynakları ve sağladıkları başarımın bilinmesi, kullanıcı isteklerinin daha verimli karşılanabilmesi ve kaynak planlamasının yapılması için gereklidir. Bu amaçla bildirimizin ilk katkısı, son üç yıldaki literatürden sık kullanılan donanım hızlandırıcı gerçekleştirimlerinin (matris çarpımı, yüz algılama, FFT) özelliklerini ortak parametre ve metrikler üzerinden derlemektir. Sunduğumuz sayısal değerler bulut kaynak ataması ve örnek bulut iş yüklerinin yaratılabilmesi için kullanılabilir. Bildirimizin ikinci katkısı açık kaynak kodlu ve HLS (High-level Synthesis-üst seviye sentezleme) ile gerçeklenmiş bir kütüphane kullanarak örnek bir donanım hızlandırıcısı olan Canny kenar algılayıcısının gerçeklenmesidir. Bu şekilde donanım hızlandırıcı gerçeklerken izlenebilecek iş akışı ve ölçülen başarım sunulmaktadır.**

Anahtar Kelimeler—donanım hızlandırıcı, Canny kenar algılama, FPGA, bulut bilişim.

***Abstract*—In this paper, hardware accelerator (FHA) applications realized on FPGA that can be offered as a service in cloud computing systems are discussed. It is necessary to know the hardware resources used by FHA applications and the performance they provide for the efficient meeting of the user requests and effective resource planning. To this end, the first contribution of this paper is to provide a compilation of the literature on the features of frequently used hardware accelerators (matrix multiplication, face detection, FFT) in the last three years, based on common parameters and metrics. The numerical values we provide can be used for cloud resource allocation and creation of sample cloud workloads. The second contribution of our paper is the implementation of the Canny edge detector, a sample hardware accelerator implemented in HLS (High-level Synthesis), using an open source library. In this way, the work flow for the implementation and operation of the hardware accelerator together with its performance are presented.**

Keywords—hardware accelerator, Canny edge detection, FPGA, cloud computing.

I. GİRİŞ

FPGA üzerinde gerçeklenen donanım hızlandırıcılar (FDH) uygulamaya özel tasarlanırlar. Yeniden yapılandırılabilir olmaları ve yapısal paralel çalışmaları ile işlemci gerçekleştirimlerine göre yüksek iş çıktısı (throughput) sağlarlar. Kapsamı sürekli gelişen bulut bilişim servisleri kapsamında donanım hızlandırıcıların da kullanıcıya sunulabilmesi için FPGA sanallaştırma teknikleri geliştirilmiştir [1]. [2] çalışmasında özellikle hesaplama ağırlıklı uygulamalar için donanım hızlandırıcıların işlemci gerçekleştirimlerine göre onlarca kat fazla çıktı başarımı sağladıkları görülmüştür. Bu başarım oranı donanım hızlandırıcının veriyi işlemek için işlemciyle ya da dış dünyayla haberleşmesini gerektiren akan veri uygulamalarında azalmaktadır.

FDH'ler bağlı bulundukları bulut sunucusundaki işlemci tarafından gönderilen hesaplama görevleri sonuçlarını işlemciye geri iletirler. FPGA üzerinde gerçeklenen donanım hızlandırıcılar doğrudan kullanıcıya IaaS (Infra Structure as a Service - Hizmet olarak Altyapı) olarak ya da SaaS (Software as a Service - Hizmet olarak Yazılım) kapsamında gelen veriyi işleyecek işlemciye alternatif platform olarak kullanılabilirler. Bu servisler bulut veri merkezinde mevcut yazılım ve donanım kaynaklarının isteklere atanması ile gerçekleştirilir.

IaaS kapsamında kaynak ataması yapılırken kullanıcı tarafından özellikleri tanımlanan donanım hızlandırıcıya uygun sanallaştırma tekniği ile gerekli donanım kaynağı miktarının belirlenmesi gerekmektedir. SaaS kapsamında ise kullanıcı girdi verisinin özellikleri (görüntü çözünürlüğü, matris boyutları) ve kullanıcı başarım gereklerine (çıktı-throughput) göre uygulamanın koşturulacağı uygun platform (işlemci ya da belli özelliklerde donanım hızlandırıcı) belirlenmelidir. Donanım hızlandırıcılı bulut veri merkezlerinin kaynak yönetiminde kullanıcı isteklerini karşılayabilecek gerçekleştirim alternatiflerini göz önüne alan [3] gibi yöntemlerde farklı girdi boyutları için donanım hızlandırıcı gerçekleştirimlerinin kaynak ihtiyaçlarının ve başarımlarının bilinmesi gerekmektedir.

Bulut veri merkezlerinin düzenli olarak kapasitelerinin güncellenmesi ve planlanması gerekmektedir. Bu nedenle farklı bulut sunucu ve donanım hızlandırıcı konfigürasyonlarının

978-1-6654-3649-6/21/$31.00 ©2021 IEEE

2021 29th Signal Processing and Communications Applications Conference (SIU) | 978-1-6654-3649-6/21/$31.00 ©2021 IEEE | DOI: 10.1109/SIU53274.2021.9478015

farklı iş yükleri altında başarımlarının bilinmesi gerekmektedir. Bu kapsamda mevcut sanal makine izlerinin özelliklerini modelleyerek, parametreleri ayarlanabilir model tabanlı gerçekçi iş yükleri yaratılması önemlidir [4]. Sanal makine istek izleri ya da SaaS istek izleri literatürde mevcut olmasına karşılık donanım hızlandırıcıları kapsayan bir örnek istek izi bulunmamaktadır.

Bu bildirinin ilk katkısı, sık kullanılan uygulamaların *donanım hızlandırıcı gerçekleştirimlerinin sistematik bir sunumudur*. Bu amaçla literatüde son üç yılda yapılan çalışmalar ile birlikte yeni bir açık kaynak hızlandırıcı kütüphanesi olan Xilinx Vitis [5] incelenmiştir. Seçilen gerçekleştirimler arasında yeni nesil bulut bilişim veri merkezleri için donanım hızlandırıcı geliştirme amacıyla özel tasarlanan FPGAler de mevcuttur. Çalışmalardaki ortak parametreler belirlenmiş, hızlandırıcı gerçekleştirimleri kaynak kullanımı ve başarım sayıları ile düzenlenmiştir. Sunulan sayısal veriler donanım hızlandırıcılı bulut bilişim sistemleri için model tabanlı iş yükü üretimi ve kaynak atanması yöntemlerinde kullanılabilecektir.

Bildirinin ikinci katkısı bulut bilişim servisi uygulaması olabilecek bir *örnek donanım hızlandırıcının gerçekleştirimi ve başarım değerlendirilmesidir*. Bu kapsamda Xilix Vitis kütüphanesinde HLS (High-level Synthesis-üst seviye sentezleme) diliyle gerçeklenen Canny kenar algılayıcısının gerçekleştirim ve çalışma iş akışı ile elde edilen başarım verileri detaylı olarak sunulmuştur.

II. Önceki Çalışmalar

Donanım hızlandırıcılı bulut bilişim sistemlerinde sanallaştırma FPGA kaynaklarının yeniden yapılandırılabilir bölgeler (YYB) olarak sunulması ve uygulamanın birden fazla FPGA üzerinde dağıtılması olarak yapılabilir [1], [6]. YYBler ile yapılan sanallaştırma daha esnek kaynak yönetimi sağlasa da FPGA'in bir bölümünün statik donanım için kullanılması gerekmektedir. Sanallaştırma yazılımda uyarlanmış hipervizör ya da hipervizör görevi gören statik donanımla yapılır. Hipervizör PCIe üzerinden erişilen FPGA kartını işlemciye çevre birim olarak gösterir.

Yakın tarihli FDH çalışmaları arasında, [2] farklı global bulut servislerinin donanım hızlandırıcılarının başarımını işlemciye göreceli olarak ve kullanıcı bakış açısıyla olarak incelemiş, somut FPGA kaynak kullanımını belirtmemiştir. [7], bir benchmark gerçeklemiş ve iki farklı platformda başarım incelemesi yapmıştır. Günümüzde gittikçe daha yaygın kullanılan doğrudan bulut bilişim için hızlandırıcı geliştirme amaçlı yeni nesil FPGA'ler [8] bu çalışmada ele alınmamıştır. [9]'da farklı kartlar üzerinde hızlandırıcı gerçekleştirimi yapılmıştır. Ele alınan uygulamaların bazıları gerçek uygulamalar değildir sadece ölçüm amaçlıdır. Bu çalışmamızda [9]'da sunulan matris çarpım uygulaması da incelenmektedir.

Çalışmamızda gerçeklediğimiz Canny kenar algılamasının alındığı Vitis kütüphanesi Xilinx tarafından sunulmakta olup FPGA tasarım sürecini kolaylaştırmayı ve hızlandırmayı amaçlamaktadır [5], [10]. Kütüphanenin yeniliği nedeni ile literatürde Vitis kullanılarak yapılan gerçekleştirmeler sınırlıdır.

III. Seçilen Donanım Hızlandırıcı Uygulamaların Profilleri

Bu çalışmamızda matris çarpımı, yüz algılama ve FFT olmak üzere üç uygulamayı incelemekte ve derlediğimiz sayısal verileri Tablo I'da sunmaktayız.

Matris Çarpımı: [9]'da ele alınan matris çarpımı [11]'de kullanılan tek duyarlı kayan nokta (single precision floating point) veri tipinde 4096x4096 boyutundaki matrisleri 8x8 bloklar olarak çarpmaktadır. Kullanılan hızlandırıcı çekirdek içinde 3 boru hattı bulunmaktadır. DDR ve HBM2 hafıza tiplerini yoğun olarak kullanan iki farklı gerçekleştirim yapılmıştır.

HBM2 ağırlıklı gerçekleştirim kullanılan hafıza daha hızlı olduğu için daha yüksek başarım göstermektedir. Çıktı başarımı frekansla orantılı bir şekilde artmıştır.

Yüz algılama: [7]'de Viola Jones algoritması ile 320x240 gri ölçek görüntülerden yüz algılaması yapılmaktadır. Kullanılan Viola Jones ardışık sınıflandırıcı (Cascade classifier) başarımı önemli derecede etkilemektedir. Mevcut kaynaklar bütün sınıflandırıcıları paralel gerçeklemeye yetmemektedir. En çok çağırılan ilk üç aşama paralel ve veriyi yazmaçlarda tutarak gerçekleştirilmiştir. Kalan aşamalar boru hattı yapılarak çıktı hızı artırılmıştır. Görüntü her saat döngüsünde alt pencerelere bölünerek işlenmiştir. [12]'de kullanılan MTCNN girdi ve çıktıları ardışık üç farklı sinir ağını kullanmaktadır. Kullanılan paralelleştirme yönteminde girdi ve sinir ağı ağırlık değerleri ping-pong yöntemi ile çalışan farklı iki arabellekte tutularak hafıza erişim gecikmesi azaltılmaktadır. Hesaplama zamanı girdi görüntü içindeki yüz sayısı ve büyüklüğü ile değişmektedir.

Raporlanan sonuçlarda Amazon Web Services (AWS) F1 tipinde kullanılan UltraScale+ VU9P platformu, Xilinx ZC706 XC7Z045 platformundan daha yüksek başarım elde etmektedir.

FFT: [9]'da 1 boyutlu farklı büyüklüklerde FFT toplam 2 GB veri için boru hattı konfigürasyonu ile hesaplanmaktadır. Buna göre her saat çeviriminde global hafızaya sekiz değer yazılarak FFT büyüklüğünün (S) logaritması kadar karmaşık kayan nokta çarpma yapılmaktadır. Birden fazla çekirdekle hesap yapılabilmektedir. Her çekirdek başına iki hafıza bankası gerekmektedir. Kullanılan kaydıran yazmaç boyutu başarımı etkilemektedir. [13] DSP kütüphanesi tamamen sentezlenebilir Super Örnek Veri Hızı (Super Sample data Rate -SSR) FFT gerçeklemesi yapmaktadır. Kullanılan sistolik mimari birden fazla örneği tek saat çeviriminde işlemektedir. Bir saat çeviriminde işlenen örnek sayısı (P) SSR faktörüne bağlıdır. FFT akan veri işleyen bir yapıya sentezlenen C++ şablon fonksiyonu olarak gerçeklenmektedir.

[9]'da aynı 32 büyüklüğünde FFT için HBM2 ağırlıklı yapılan gerçekleştirimde 15 çekirdek kopyası kullanılarak çıktı başarımı tek çekirdek kopyası olan DDR ağırlıklı gerçeklemeye göre 7 kattan fazla artmıştır. [13] gerçekleştirimleri farklı P değerleri için 1024 ve 4096 FFT büyüklükleri için yapılmıştır. Burada P, 2'den 8'e çıktığında frekans değeri fazla azalmamış buna karşılık artan paralel işlem miktarı ile çıktı başarımı çok yükselmiştir. P değeri 16'ya çıktığında frekans çok düştüğü için çıktı başarımı neredeyse aynı kalmıştır.

TABLO I: Seçilen Donanım Hızlandırıcıların Özellikleri

Yıl, Referans	Platform	Girdi Veri	Kaynak Kullanımı (LUT, FF: x1000)	Çıktı Başarımı	Frekans
Matris çarpımı					
2020, [9]	Xilinx Alveo U280 DDR Xilinx Alveo U280 HBM2	4096 × 4096 matris, Blok boyutu:8 [11]	LUT(569), FF(442), BRAM(666), DSP(7,683) LUT(499), FF(920), BRAM(666), DSP(7,683)	266.9 GFLOP/s 603.9 GFLOP/s	100 MHz 236 MHz
Yüz Algılama					
2018, [7]	Xilinx ZC706 AWS F1 Xilinx Virtex UltraScale+ VU9P	320x240, gri ölçek, Viola Jones	LUT(63), FF(84), BRAM(121), DSP(79) LUT(48), FF(54), BRAM(92), DSP(72)	30.3 frames/s 46.5 frames/s	140 MHz 250 MHz
2019, [12]	Xilinx ZC706	Yüksek çözünürlük, renkli, MTCNN algoritması [14]	LUT(134), FF(222), BRAM(196), DSP(880)	11.9 frames/s	200 MHz
1 Boyutlu Kayan Nokta FFT (S: FFT büyüklüğü (FFT size), K: Çekirdek Sayısı P: Paralel işlenen örnek sayısı)					
2020, [9]	Xilinx Alveo U280 DDR Xilinx Alveo U280 HBM2	K: 1, S:512 [11] K: 15, S:32 [11]	LUT(83), FF(168), BRAM(39), DSP(672) LUT(602), FF(941), BRAM(405), DSP(5,280)	78.3 GFLOP/s 576.2 GFLOP/s	248 MHz 254 MHz
2020, [13]	Xilinx Alveo U250	P:2, S:1024 P:2, S:4096 P:8, S:1024 P:8, S:4096 P:16, S:1024 P:16, S:4096	LUT(45.3), FF(91.3), BRAM(24), DSP(554) LUT(66), FF(110), BRAM(50), DSP(670) LUT(129.8), FF(247), BRAM(52), DSP(1424) LUT(158), FF(268), BRAM(78), DSP(1606) LUT(283), FF(520), BRAM(91), DSP(3072) LUT(380), FF(592), BRAM(46), DSP(3664)	49.9 GFLOP/s 57.1 GFLOP/s 145.9 GFLOP/s 154.5 GFLOP/s 151.4 GFLOP/s 95.6 GFLOP/s	500 MHz 476 MHz 367 MHz 322 MHz 192 MHz 100 MHz

IV. Örnek Donanım Hızlandırıcı Uygulama Gerçekleştirimi: Canny Kenar Algılama

Bu çalışmamızda, örnek donanım hızlandırıcı gerçekleştirimi iş akışını göstermek amacıyla Canny kenar algılama (Canny Edge Detection-CED) uygulamasını Xilinx ZC706 platformunda gerçekleştirdik ve başarımını ölçtük. Gerçekleştirdiğimiz CED, HLS dilinde, açık kaynak kodlu olarak [15]'de bulunan koddan türetilmiştir.

CED uygulaması, ZC706 platformunda bulunan ARM işlemcisi üzerinde koşacak ana kod (AK) ve FPGA alanı üzerinde koşacak mantıksal donanım tasarımı olmak üzere iki akış ile derlenerek oluşturulmuş bir yongada sistem uygulamasıdır. ARM üzerinde koşacak C kodu ile geliştirilmiş uygulama "arm-linux-gnueabihf-g++" çapraz derleyicisi ile derlenmişir. FPGA alanında koşacak olan donanımlar Vitis v++ derleyicisi ile oluşturulmuştur. Diğer donanım hızlandırıcılarda olduğu gibi veri akışının kontrolü işlemcide çalışan uygulama tarafında yapılırken hesaplama işlemleri FPGA üzerinde yapılmaktadır.

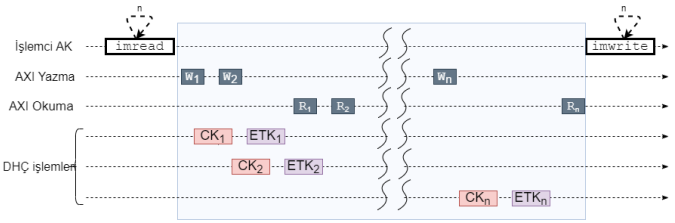
CED Bölüm III'deki hızlandırıcılardan farklı olarak iki alt çekirdekten oluşmaktadır. Bu özelliği ile farklı FPGA'lere ya da FPGA YYB'lerine dağıtılarak daha ölçeklenebilir bulut servislerine imkan tanıyabilecek bir gerçekleştirime olanak sağlamaktadır.

CED uygulaması, FPGA üzerinde koşan iki donanım hızlandırıcı çekirdek (DHÇ) ile gerçekleştirilir. Canny Kernel (CK) kenar tespiti yaparken, Edge Tracer Kernel (ETK) ise güçlü pikseller arası bağlantının çizilmesi işlemini gerçekleştirir. Girdi görüntüye (GG) kenar tespiti için 3x3 Gaussian maskesi ve Sobel filtre uygulanır. İki yönlü gradyan hesabı ile bulunan zayıf pikseller "Non-maximal suppression" kullanılarak görüntüden kaldırılır. Daha sonra ETK'de, güçlü piksellerden oluşan kenarların bağlantısı tamamlanır.

CK ve ETK birbirleri ve işlemci üzerinde koşan AK ile AXI arayüzleri üzerinden OpenCL ile Şekil 1'de görüldüğü gibi haberleşerek çalışır.

AK, OpenCV kütüphanesinin yazma ve okuma fonksiyonlarını kullanarak sabit diskten işlenecek olan n adet görüntüyü `imread` ile bir matris dizisine okur ve DHÇlerde işlenen görüntüler yine işlemci global hafızasına bir matris dizisi olarak yazıldıktan sonra sabit diske `imwrite` yöntemi ile aktarılır.

Matris dizisindeki her girdi görüntü $GG_i, i = 1, \cdots, n$ AK tarafından $\texttt{W}_i$ ile AXI Stream arayüzünden FPGA'in global hafızasına yazılır. AK global hafızada GG_i başlangıç adresi, çıktı görüntü CG_i başlangıç adresi, görüntü boyutu ve filtre parametrelerini AXI Lite arayüzünden DHÇ'lere iletir. AK, AXI Lite arayüzünden DHÇ'leri çalışmaya başlamaları için tetikler. CK global hafızada yer alan GG_i'yi işler ve ürettiği GG_i^{CK}'yi AXI Stream arayüzünden ETK'ye işlemesi için iletir. ETK işlemeden sonra ürettiği CG_i'yi global hafızaya yazar ve işlemin tamamlandığını AXI Lite arayüzünden AK'a iletir. AK, CG_i'yi FPGA'in global hafızasından $\texttt{R}_i$ ile okuyarak işlemci global hafızasına çeker.



Şekil 1: İş akışı.

Tablo II'de görüldüğü gibi CED'nin kaynak kullanımı düşük bulunmuştur. Gerçekleştirim 512x512 renkli JPEG görüntüleri işlemektedir ve 142 MHz'de çalışmaktadır. Global hafıza alanları arasındaki transfer süreleri işlemci'den FPGA tarafına 1.845 ms, FPGA'den işlemci tarafına 1.934 ms olarak ölçülmüştür. Vivado'nun güç analizi aracından elde ettiğimiz sonuçlarda, toplam tüketimi 2.162 W olan sistem çipinin, güç tüketiminin büyük çoğunluğunun 1.576 W ile FPGA üzerindeki ARM işlemci tarafından yapıldığı gözlemlenmiştir.

V. Sonuç ve Gelecekteki Çalışmalar

Bu bildiride bulut servisleri için FPGA yeniden yapılandırılabilir bölgelerinde gerçeklenen donanım hızlandırıcılar ele

TABLO II: CED Donanım Hızlandırıcı Gerçeklemesi

DHÇ	LUT	BRAM	DSP	Güç	Koşma zamanı
Canny	6,800	9	10	0.060 W	4.647 ms
Edge Tracing	5,655	14	9	0.054 W	2.578 ms

alınmaktadır. Bu kapsamda literatürde son üç yılda yayınlanmış, sık kullanılan uygulamaların FPGA üzerinde donanım hızlandırıcı gerçekleştirimleri incelenmiştir. Ortak parametreler ve metrikler üzerinden sonuçları olan; matris çarpımı, yüz tanıma ve FFT uygulamaları seçilmiştir. Bu uygulamaların farklı donanım platformları üzerinde ve farklı veri girdileri ile kaynak kullanımı, çıktı başarımı ve frekans değerleri düzenlenerek sunulmuştur. Gözlemlenen sonuçlarda, hızlı hafıza tipi olan HBM2 kullanılan gerçekleştirimlerin daha yüksek başarım elde ettiği gözlemlenmiştir. Donanım paralelleştirmesi arttıkça ilk aşamalarda tasarımın kaynak kullanımı artmasına rağmen frekansın az etkilenmesi nedeni ile çıktı başarımının arttığı fakat belli bir noktadan sonra frekans fazla düştüğü için paralelleştirmeden yararlanılamadığı görülmüştür. Düzenlediğimiz veriler bulut kaynak planlamasında ve iş yükü modellemesinde kullanılabilecektir.

Donanım hızlandırıcı gerçekleştirimi ve işleyişini detaylı olarak gösterebilmek için açık kaynak bir kütüphane kullanılarak Canny kenar algılama uygulaması gerçeklenmiştir. Bu uygulama çift hızlandırıcı çekirdeği ve işlemcinin birlikte çalışmasını gerektirdiği için bulut bilişim donanım sanallaştırması kapsamında dağıtılabilir örnek bir uygulamadır. Başarım ölçümünde, FPGA kaynaklarının fazla tüketilmediği görülmüştür. Güç tüketiminin başlıca büyük bir bölümünü işlemcinin yaptığı da gözlenmiştir.

Gelecek çalışmalarımızda sunduğumuz hızlandırıcı verileri kullanılarak [4] daki iş yükü modelleri donanım hızlandırıcılar ile genişletilecek ve [3] kaynak atama yöntemi daha gerçekçi girdilerle değerlendirilecektir. Laboratuvar ortamında [16] kurduğumuz donanım hızlandırıcılı ve ağ bağlantılı bulut sunucuları üzerinde donanım hızlandırıcıların farklı bulut servisleri olarak sunulması gerçekleştirilecek ve farklı dağıtılmış FPGA hızlandırıcılarla sunduğumuz parametre ve metriklere güç ve uygulama gecikmesi de eklenerek detaylı ölçüm yapılacaktır.

TEŞEKKÜR

Bu çalışma, 117E667-117E668 nolu proje kapsamında TÜBİTAK tarafından desteklenmektedir. Yazarlar desteklerinden dolayı TÜBİTAK'a ve ASELSAN A.Ş.'ye teşekkür eder.

KAYNAKLAR

[1] L. M. Al Qassem, T. Stouraitis, E. Damiani, and I. A. M. Elfadel, "Fpgaaas: A survey of infrastructures and systems," *IEEE Transactions on Services Computing*, 2020.

[2] X. Wang, Y. Niu, F. Liu, and Z. Xu, "When fpga meets cloud: A first look at performance," *IEEE Transactions on Cloud Computing*, 2020.

[3] N. U. Ekici, K. W. Schmidt, A. Yazar, and E. G. Schmidt, "Resource allocation for minimized power consumption in hardware accelerated clouds," in *2019 28th International Conference on Computer Communication and Networks (ICCCN)*. IEEE, 2019, pp. 1–8.

[4] F. Koltuk and E. G. Schmidt, "A novel method for the synthetic generation of non-iid workloads for cloud data centers," in *2020 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 2020, pp. 1–6.

[5] "Xilinx Vitis," https://www.xilinx.com/products/design-tools/vitis/vitis-platform.html, 2021.

[6] A. M. Caulfield, E. S. Chung, A. Putnam, H. Angepat, J. Fowers, M. Haselman, S. Heil, M. Humphrey, P. Kaur, J.-Y. Kim *et al.*, "A cloud-scale acceleration architecture," in *The 49th Annual IEEE/ACM International Symposium on Microarchitecture*. IEEE Press, 2016, p. 7.

[7] Y. Zhou, U. Gupta, S. Dai, R. Zhao, N. Srivastava, H. Jin, J. Featherston, Y.-H. Lai, G. Liu, G. A. Velasquez *et al.*, "Rosetta: A realistic high-level synthesis benchmark suite for software programmable fpgas," in *Proceedings of the 2018 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2018, pp. 269–278.

[8] "Xilinx Alveo," https://www.xilinx.com/products/boards-and-kits/alveo.html, 2021.

[9] M. Meyer, T. Kenter, and C. Plessl, "Evaluating FPGA accelerator performance with a parameterized OpenCL adaptation of selected benchmarks of the HPCChallenge benchmark suite," in *2020 IEEE/ACM International Workshop on Heterogeneous High-performance Reconfigurable Computing (H2RC)*. IEEE, 2020, pp. 10–18.

[10] N. Brown, "Weighing up the new kid on the block: Impressions of using vitis for hpc software development," in *2020 30th International Conference on Field-Programmable Logic and Applications (FPL)*. IEEE, 2020, pp. 335–340.

[11] P. Luszczek, J. J. Dongarra, D. Koester, R. Rabenseifner, B. Lucas, J. Kepner, J. McCalpin, D. Bailey, and D. Takahashi, "Introduction to the HPC challenge benchmark suite," Ernest Orlando Lawrence Berkeley NationalLaboratory, Berkeley, CA (US), Tech. Rep., 2005.

[12] C. Fu and Y. Yu, "Fpga-based power efficient face detection for mobile robots," in *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2019, pp. 467–473.

[13] "Xilinx Vitis 1-Dimensional(line) FFT L1 FPGA module," https://xilinx.github.io/Vitis_Libraries/dsp/2020.2/user_guide/L1.html#l1-performance-benchmarks-and-qor, 2021.

[14] S. Yang, P. Luo, C.-C. Loy, and X. Tang, "Wider face: A face detection benchmark," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 5525–5533.

[15] "Xilinx Vitis Canny Edge," https://github.com/Xilinx/Vitis_Libraries/tree/master/vision/L2/examples/canny, 2021.

[16] E. G. Schmidt and A. Yazar, "A novel fpga accelerated cloud architecture." [Online]. Available: http://accloud.eee.metu.edu.tr/