Otomatik Kod Üretimi ile Sinyal İşleme Uygulamalarında Yeniden Kullanılabilirliğinin Artırımı ve Yazılım Geliştirme Süresinin Azaltılması
The Improvement of Reusability and Reduction of Software Development Period with Automatic Code Generation in Signal Processing Applications

Alparslan Fişne[0000−0003−2120−9260]

Aselsan A.Ş., REHİS Sektör Başkanlığı, Ankara, Türkiye
afisne@aselsan.com.tr

Özet: Yazılım uygulaması tasarımında algoritma tasarımı ve yazılım geliştirme süreci, zaman bağımlılığı olan süreçlerdir. Algoritma tasarım sürecinde, algoritma tasarım mühendisleri uygulamanın algoritmalarını hazırlar. Algoritmaların hazırlanmasından sonra yazılım geliştirmeciler kodlama çalışmalarına başlar. MATLAB gibi model geliştirme araçlarında hazırlanan algoritmalar için kodlama öncesinde yazılım geliştiricilerin algoritmaları öğrenme ve anlama süreci yazılım geliştirme süresini uzatır. Bu çalışmada, otomatik kod üretim aracı ile algoritmalardan C++ kod üretilerek yazılım geliştirme süresinin azaltılması amaçlanmaktadır. Ayrıca MATLAB Kodlayıcı kullanımı ile elde edilen kazanımlar gösterilmektedir.
Anahtar Kelimeler: otomatik kod üretimi, yeniden kullanılabilirlik, MATLAB Kodlayıcı.

Abstract: The algorithm design and software development process are time-dependent processes in software application design. In the algorithm design process, algorithm design engineers set up the algorithms of the application. After the preparation of the algorithms, the software developers begin coding studies. Before coding, the process of learning and understanding algorithms of software developers for algorithms which are prepared in model development tools such as MATLAB extends software implementation period. In this study, it is purposed that software developing period are reduced by generating C++ code from algorithms. Also it is shown that outcomes are obtained by usage of MATLAB Coder.
Keywords: automatic code generation, reusability, MATLAB Coder.

1 Giriş

Gömülü yazılımların yer aldığı çözüm mimarilerinde içerilen bileşenler ve yetenekler ile sistem karmaşıklığı gün geçtikçe artmaktadır. Karmaşıklığın azaltılması ve bileşenler ile modüler bir mimari oluşturulması için işlevlerdeki soyutlama katmanının doğru yöneltilmesi gerekir. Ayrıca proje teslimat tabanlı sistem üretilen alanlarda yazılım geliştirme sürelerin azaltılması ve teslimat öncesi ön ürünlere hızlı bir çözüm üretilmesi önemli bir ihtiyaç olarak karşımıza çıkmaktadır. Sistem, donanım ve yazılım ortak tasarımı ile otomatik kod üretimi için akademi ve gömülü yazılım endüstrisinde çalışmalar yapılmaktadır [1]. Otomatik kod üretimi öncesi yapılan en önemli işlemlerden biri farklı tasarım ailelerinde çalışan tasarımcıların ortak yazılım tasarımı için gereklerin doğru belirlemesidir [2]. Bu sistematik akış için MATLAB, Simulink vb. araçlar kullanılmaktadır [3]. Otomatik kod üretimi yapan araçlarda donanıma özgü özellikleri kullanıma açan opsiyonlar bulunmaktadır. Günümüz bilgisayar mimarileri, özellikle de CPU, GPU, DSP ve FPGA gibi gelişmiş bilgisayar mimarileri, kullanıcılara donanım özgü işler yapılabilmesi için donanıma erişen yazılım katmanları ve fonksiyonları sunmaktadır. Otomatik kod üretimi sayesinde C, C++, VHDL gibi diller ile uzun süren yazılım geliştirmeler yerine hazır kaynak kodları ve modeller elde edilebilmektedir [4], [5]. Bu sayede yazılım geliştirme sırasında harcanacak sistem öğrenme ve anlama süresi büyük bir oranda kısaltılır. Savunma sanayi endüstrisinde sistem mühendisleri, algoritma ve sistem senaryo tasarımı yaptıktan sonra ürün modellemesi için tasarım mühendislerine çalışmalarını aktarırlar. Aktarılan çalışma doğrultusunda yazılım geliştiriciler tasarım çalışmaları için yol haritası belirlemeye başlar. Bu aktarım sürecinde öğrenme ve anlama süresi, kişinin bilgi ve becerileri ve alan bilgisine bağlı olarak değişebilmekte, sonuç olarak bir süre harcanmasına neden olmaktadır.

Bu çalışmamızda, sistem ve yazılım tasarımcılarının daha verimli çalışabilmesi için sistem algoritmasından otomatik kod üretimi ile yazılım yapıtaşlarının elde edilmesini çalıştık. Otomatik kod üretimi çalışmalarında sinyal işleme uygulamalarının MATLAB Kodlayıcı ile hazırlanarak tasarlanması odak noktamız olmuştur. Otomatik kod üretimi ile elde edilen yazılım bileşenleri ile elle yazılan kodlarla oluşturulmuş yazılım bileşenleri arasındaki işlem performans sonuçları ve yazılım geliştirme maliyetleri karşılaştırmalı olarak sunulmaktadır. Bu çalışma beş ana bölümden oluşur. Bu makalenin ikinci bölümünde otomatik kod üretimi ile ilgili yapılan çalışmalar hakkında bilgi verilmektedir. Üçüncü bölümde sinyal işleme yazılım yapıtaşı bileşenlerinin MATLAB Kodlayıcı ile üretilmesi ve yazılım yeniden kullanılabilirlik için yapılan çalışmalar sunulmaktadır. Dördüncü bölümde otomatik kod üretimi ile elde edilen kazanımlar gösterilmektedir. Beşinci bölümde otomatik kod üretiminin sağladığı avantajlar ve gelecekte yapılacak çalışmalar hakkında bilgi verilmektedir.

2 Literatür Çalışması

Otomatik kod üretimi, algoritma modelleme ve tasarım geliştirme çalışmaları yapan firmalarda son yıllarda önemi giderek artmaktadır. Otomatik kod üretimi

sonucu üretilen yazılımlar gömülü ortamlarda da kullanılmasının yanı sıra Windows, Linux gibi işletim sistemlerinde masaüstü uygulaması olarak sunulabilmektedir. Otomatik kod üretimi sadece C/C++ dilleri ile sınırlı kalmayıp Kullanıcı Arayüz (GUI) Yazılımları için gerekebilecek JAVA tabanlı .jar dosyalarının da üretilmesini sağlayabilmektedir [6]. Donanım tasarım geliştirme aşamasında özellikle VHDL dili ile yapılan gerçekleştirmelerde yaşanan tasarım geliştirme maliyetlerinden dolayı otomatik kod geliştirme ve yüksek seviyeli sentezleme kullanımı, tasarım geliştirme sürelerini iyileştirmeye başlamıştır [7].

Otomatik kod üretimi model tabanlı tasarımlar sayesinde de yapılmaktadır. Model tabanlı tasarımlar UML ile yapılarak nesneler arasındaki ilişkiler ve nesne tabanlı tasarımlar, otomatik kod üretimi sayesinde oluşturulabilmektedir. Moreria ve arkadaşlarının yaptığı çalışmada gömülü sistemler için UML tasarımlardan VHDL koduna geçiş anlatılmaktadır [8]. Bu çalışma VHDL ile geliştirmeyi daha soyut hale getirmiş olup kullanıcıya bakım ve sürdürebilirlik faktörleri açısından iyileştirme sağlamıştır.

Otomatik kod üretimi ayrıca optimizasyon tekniği olarak kullanılabilmektedir. Çok çekirdekli işlemcilerde OpenMP ile veri paralelleştirme yapılabilmektedir. OpenMP direktifleri kullanılarak paralel çalışan kod parçaları üretilebilmektedir. Membarth ve arkadaşları CPU ve GPU ortamlarında medikal görüntüleme uygulaması için otomatik kod üretim direktifleri kullanmışlardır [9]. Bu çalışma yazılım geliştiricilerinin optimizasyon ihtiyacını karşılayan yöntemleri çalışan ve aktaran bir gösterim sunmaktadır.

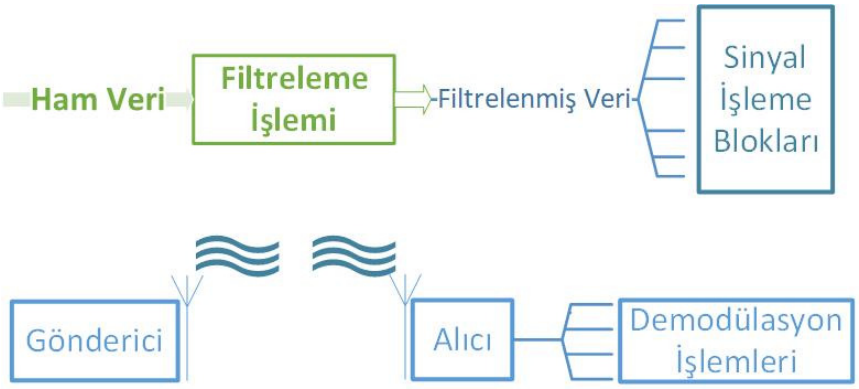
Web tabanlı uygulamalarda son kullanıcıya yönelik otomatik kod üretim ve model geliştirme araçları son zamanlarda rağbet gören bir geliştirmedir. Akbulut ve arkadaşları, görsel programlama tabanlı son kullanıcıya yönelik otomatik kod üretim aracını tasarlamışlardır [10]. Kullanıcıdan web tecrübesi aranmadan görsel programlama ile otomatik uygulama oluşturulabilmektedir. Çalışmanın sonucunda otomatik kod üretimi için jenerik yaklaşımın yerine alana özel tasarım yaklaşımı olması gerektiği söylenmektedir.

3 MATLAB Kodlayıcı ile Uygulamaların Gerçekleştirimi

Bu çalışmada sinyal işleme yazılım konfigürasyon birimlerinde yer alan temel iki uygulamadan bahsedilmektedir.

3.1 Sinyal İşleme Uygulamaları

Bu çalışmada sinyal işleme uygulaması olarak filtreleme yeteneği çalışılmaktadır. Ayrıca demodülasyon işlemleri için karşılaştırmalar yapılmaktadır. Filtreleme sinyal işleme yazılımlarında ön blok olarak çoğu zaman karşımıza çıkmaktadır [11]. Ayrıca gönderici tarafından modülasyonu yapılmış sinyallerin içinde yer alan mesajların alıcı tarafta çözülmesi için uygulanan demodülasyon işlemleri çok sayıda alt sinyal işleme blokları içermektedir [12]. Şekil 1'de çalışılan yeteneklere ait şemalar gösterilmektedir.

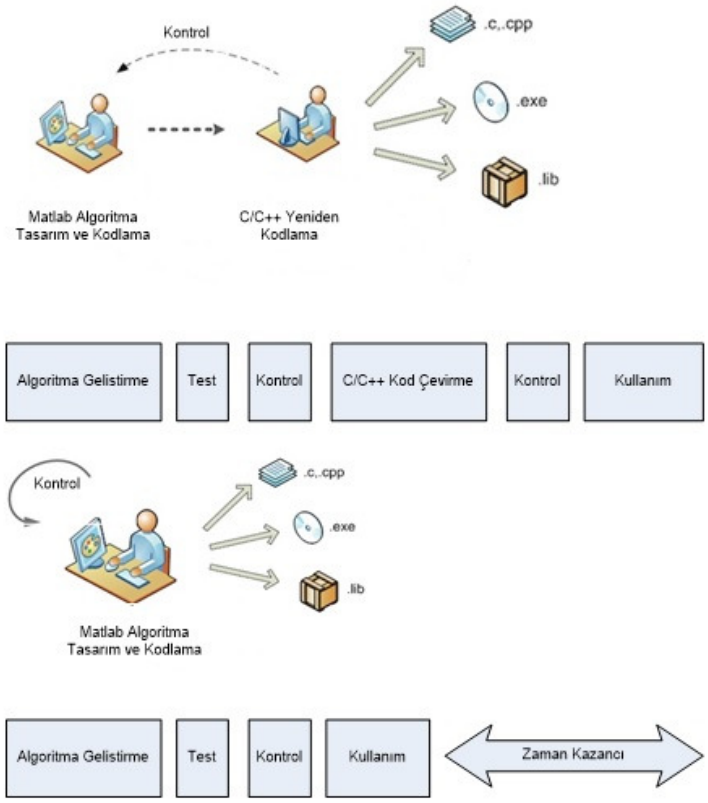


Şekil 1. Sinyal İşleme ve Demodülasyon Adımları

3.2 MATLAB Kodlayıcı Kullanımı

MathWorks firması özellikle donanım geliştirme uygulamaları için Simulink modelden HDL koduna çevrim yapan HDL Kodlayıcı aracını önceden beri sunmaktadır [13]. Özellikle 2013 yılından sonraki MATLAB versiyonlarında C/C++ kodlayıcı aracı olan MATLAB Kodlayıcı yazılım geliştiricilere büyük kolaylık sağlamaktadır [14]. Gömülü ortamlar için çalıştırılabilir dosyası üretilmiş bir algoritma uygulamasını üretebilmek, sistem mühendislerinin test ortamlarındaki çalışmalarına katkı sağlamaktadır. Böylece sistem ve yazılım mühendisleri ortak tasarım yaparak son ürüne yönelik olan yazılım geliştirmenin süresinin azaltılması sağlarlar. MATLAB Kodlayıcı ile otomatik C++ kod üretimi yapılırken, algoritma parametre boyutlarının belirlenmesine ve bellek kullanımına dikkat edilmelidir. Üretilen her bir fonksiyon için MATLAB Kodlayıcı projesinde hem kaynak dosyası (.cpp) hem de başlık dosyası (.h) oluşturulmaktadır. Şekil 2'de bu senaryoları anlatan dikkat çekici bir anlatım bulunmaktadır.

Uygulamalar için kaynak kod jenerik üretilerek platform bağımsız uygulamaların çalıştırılması mümkündür. Bu çalışmada filtreleme ve demodülasyon işlemleri için C++ kaynak kodları otomatik olarak üretilmiştir. Filtreleme işlemi, algoritma modellerinde alt ve ön sinyal işleme adımlarında kullanılmaktadır. Ayrıca çeşitli sinyal işleme projelerinde kullanıldığı için filtreleme yazılım yapıtaşı sayesinde çok sayıda üründe yeniden kullanılabilirlik sağlanmaktadır. Filtreleme işlemi özellikle çok sayıda kompleks vektör çarpımı içerdiği için dikkatli ve optimize yazılması gerekmektedir. MATLAB Kodlayıcı sayesinde bu işlem hem hızlı hem de daha sade olmuştur. Demodülasyon işlemlerinde çok sayıda alt işlem ve ardışık akışlar olduğu için sadece yazılım tecrübesi olan geliştiricilerde öğrenme ve anlama sürecinin yaşanmasına neden olmaktadır. Otomatik kod üretimi sayesinde algoritma ve model soyutlama yeteneği, bu süreçte yazılım tasarım mühendislerine katkı sağlamaktadır. Alan bilgisi olmadan sadece yazılım tecrübesi ile demodülasyon uygulamasının gerçekleştirimi hızlandırılabilmektedir.



Şekil 2. MATLAB Kodlayıcı ile Otomatik Kod Üretimi Süreci

4 Testler ve Sonuçlar

MATLAB Kodlayıcı ile otomatik kod üretiminin getirmiş olduğu kazanımları, testler ile pekiştirmek amaçlanmıştır. Bu bölümde otomatik kod üretimi ile üretilmiş kaynak kodlardan elde edilen uygulamaların çalışma zamanı performansı ile aynı işlemler için elle kod yazılarak gerçekleştirilmiş uygulamaların performansı karşılaştırılmaktadır. Otomatik kod üretimi ile gerçekleştirilen uygulamalar için harcanan geliştirme süreleri farklı algoritmalar için aktarılmaktadır. Çalışmada kullanılan geliştirme ortamları ve araçlar Tablo 1'de verilmektedir.

Tablo 1. Testler için geliştirme ortamları

Geliştirme Ortamı	**Amaç**
MATLAB 2015b	Algoritmalar için geliştirme ortamı sağlamaktadır.
MATLAB Kodlayıcı	Algoritma modelleri C++ kodlara çevirmektedir.
Visual Studio C++	Derleme ile kodlardan uygulamalar oluşturur.

Performans ölçümlerinde test grubu olarak çeşitli algoritma fonksiyonları kullanıldı. Tablo 2'de algoritmaların elle yazılarak C++ kodu geliştirilmesi ile otomatik olarak üretilen C++ kodundan oluşturulan uygulamaların işlem performansı karşılaştırılmaktadır. Gerçek zamanlı üst sınır işlem süresi 1 sn'dir. Girdi vektör uzunluğu 1 sn'lik örnek adedidir. Algoritma performans karşılaştırmalarına bakıldığında otomatik kod üretimi hazırlanmış uygulama ile elle yazılmış C++ uygulamalarına göre hızlanma gözlemlenmektedir. Özellikle filtreleme işlemi için 7.5 kat hızlanma göze çarpmaktadır. Fakat frekans demodülasyon işleminde MATLAB Kodlayıcı kullanıldığında hızlanma yerine yavaşlama söz konusudur. Kod analiz aracı ile bakıldığında MATLAB Kodlayıcı'nın ürettiği FFT kodları jenerik tanımlama yapıldığı için işlem süresi FFTW Kütüphanesi fonksiyonlarına göre uzun sürmektedir [15]. Visual Studio C++ 2015 geliştirme ortamında ilgili kaynak kodlardaki FFT fonksiyonu FFTW fonksiyonu ile değiştirilerek frekans demodülasyondaki işlem süresi azaltılmıştır. MATLAB Kodlayıcı'dan doğrudan uygulama oluşturmak yerine üretilmiş kaynak kodlardan başka bir çalışma zamanında Visual Studio vb. ortamlarda uygulamaları oluşturmak performans faktörü açısından etkili olmaktadır.

Tablo 2. Otomatik Kod Üretimi ile İşlem Performans Kazanımları.

Algoritma	Manuel Uygulama	Otomatik Uygulama	Hızlanma
Filtreleme	24.675 ms	3.456 ms	7.5x
Genlik Demodülasyon	41.034 ms	27.631 ms	1.5x
Frekans Demodülasyon	326.754 ms	356.323 ms	0.9x
Faz Demodülasyon	18.732 ms	12.128 ms	1.5x

Sadece yazılım tecrübesi ile karmaşık algoritma modellerinin ve fonksiyonlarının alan bilgisi olmadan ürüne yönelik uygulamaları oluşturmak otomatik kod üretimi sayesinde hızlanmaktadır. Otomatik kod üretim aracı kullanılmadığı zaman uygulamayı gerçekleştirme zamanı kişiye bağlı değişkenlik göstermektedir. Tablo 3'te iki yazılım geliştiricisinin algoritmalar için tasarlamış oldukları uygulama yazılımlarının geliştirme esnasında harcadıkları zaman ve sadece yazılım tecrübesi olan bir geliştiricinin MATLAB Kodlayıcı ile geliştirilen uygulamalar için harcadığı süre karşılaştırılmaktadır. Tablo 3'te görüldüğü üzere MATLAB Kodlayıcı ile geliştirme süreleri kayda değer bir şekilde düşmektedir. Yazılım geliştirme maliyetlerinde işçilik faktörü açısından bireysel karşılaştırma ile maksimum 5 insan ay kazanç, minimum 2.5 insan ay kazanç sağlanır. Filtreleme ve demodülasyon yapıtaşları çok sayıda işlemde alt veya ön işlem olarak çağrılmaktadır. Filtreleme yazılım yapıtaşı için yeniden kullanılabilirlik dikkate alındığında 34 adet işlemde yeniden kullanılabilirliği görülmüştür. Demodülasyon yazılım yapıtaşı ise 16 adet işlemde kullanılmaktadır. Filtreleme sinyal işlemede çok kullanılan bir metot olduğu için sinyal işleme yazılımlarında bu kadar yüksek kullanımı beklenen bir durumdur. Demodülasyon işlemi yeteneğe özel bir iş olduğu için ilgili demodülasyon yeteneklerinde kullanılmaktadır.

Tablo 3. Yazılım Geliştirme Maliyetlerinin Karşılaştırılması

İşçilik Süresi	**Manuel Uygulama**		**Otomatik Uygulama**
	Geliştirici 1 (Alan bilgisi olan)	***Geliştirici 2 (Alan bilgisi az)***	***Geliştirici 3 (Yazılım tecrübesi olan)***
Filtreleme	2 hafta	4 hafta	0.5 hafta
Genlik Demod.	3 hafta	5 hafta	1 hafta
Frekans Demod.	5 hafta	8 hafta	1.5 hafta
Faz Demod.	4 hafta	7 hafta	1 hafta
Toplam İşçilik	3.5 insan ay	6 insan ay	**1 insan ay**

5 Değerlendirmeler

Otomatik kod üretimi sayesinde yazılım geliştirme maliyetleri işçilik açısından azaltılmış olup özellikle hızlı prototip ve modelleme yapılan birimlerde çeviklik arttırılmıştır. İşçilik sürelerin azalması özellikle sürekli ürün teslimatı yapan yazılım geliştiriciler için büyük fayda sağlamaktadır. MATLAB geliştirme ortamı kullanılarak tasarlanan algoritma ve modeller başka iş gücüne ihtiyaç duymadan MATLAB Kodlayıcı ile doğrudan C++ kodlarına çevrilebilmekte böylece oluşabilecek yazılım geliştirici hatasının önüne geçilmektedir. Testlerdeki bulgulara göre otomatik kod üretimi ile yapılan uygulamalar elle yazılan uygulamalara göre hızlı çalışmaktadır. Oluşturulan yapıtaşları sayesinde çeşitli alan projelerinde yeniden kullanılabilirlik ve ürün hattı yaklaşımları uygulanmaktadır. MATLAB Kodlayıcı ile yapılan tasarımların sağladığı en önemli avantaj, sistem ve yazılım mühendisliği süreci ortak yapılarak yazılım mimarisinin ürüne ait sistem mimarisinden türetilmesidir.

Jeneriklik göz önünde bulundurularak otomatik kod üretimi ile elde edilen kaynak kodları sayesinde aynı kaynak kodlar işletim sistemi bağımsız uygulamalar için kullanılabilmektedir. İşletim sistemi bağımsız yapıtaşı hazırlanarak yeniden kullanılabilirlik sağlanmış olmaktadır. Kod üretilirken bellekler dinamik açıldığı için bu durum kritik sistemlerde performansa olumsuz etki yapar. Bellek yönetimi, MATLAB modellerinde hiyerarşik bir yapıda ve yerinde bellek kullanımı ile ele alınırsa C++ kodlarına çevrimde daha az bellek alanı kullanan kod parçaları üretilebilir. Gelecekte, optimizasyon ihtiyacı olan uygulamalarda MATLAB Kodlayıcı ile kod üretilirken nasıl bir yaklaşım sergileneceği ele alınacaktır. Otomatik kod üretimi ile sistem-yazılım süreci kural haline getirilerek yeni bir yazılım mimarisi yönteminin çalışılması hedeflenmektedir.

Katkı Belirtme

Bu çalışmaya testler aşamasında katkısı olan ekip arkadaşlarım Fehmi Emre KADAN ve Batuhan Mert SEVEROĞLU'na teşekkür ederim.

Referanslar

1. Song, L., Di, L., Fang, L., Nan, Z.: CPSiCGF: A code generation framework for CPS integration modeling, Microprocessors Microsystems, v.39 n.8, p.1234-1244, November 2015. https://doi.org/10.1016/j.micpro.2015.05.010
2. Akdur, D., Garousi, V., Demirörs, O.: A survey on modeling and model-driven engineering practices in the embedded software industry, Journal of Systems Architecture,Volume 91,2018.
3. P. Banerjee et al., "A MATLAB compiler for distributed, heterogeneous, reconfigurable computing systems," Proceedings 2000 IEEE Symposium on Field-Programmable Custom Computing Machines (Cat. No.PR00871), Napa Valley, CA, USA, 2000, pp. 39-48.
4. D. de Niz, G. Bhatia and R. Rajkumar, "Model-Based Development of Embedded Systems: The SysWeaver Approach," 12th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'06), San Jose, CA, USA, 2006, pp. 231-242.
5. R. Selvamuthukumaran and R. Gupta, "Rapid prototyping of power electronics converters for photovoltaic system application using Xilinx System Generator," in IET Power Electronics, vol. 7, no. 9, pp. 2269-2278, September 2014. https://doi.org/10.1049/iet-pel.2013.0736
6. J. Sanchez, S. Dormido, R. Pastor and F. Morilla, "A Java/Matlab-based environment for remote control system laboratories: illustrated with an inverted pendulum," in IEEE Transactions on Education, vol. 47, no. 3, pp. 321-329, Aug. 2004.
7. Y. P. Siwakoti and G. E. Town, "Design of FPGA-controlled power electronics and drives using MATLAB Simulink," 2013 IEEE ECCE Asia Downunder, Melbourne, VIC, 2013, pp. 571-577.
8. T. G. Moreira, M. A. Wehrmeister, C. E. Pereira, J. Pétin and E. Levrat,: "Automatic code generation for embedded systems: From UML specifications to VHDL code,", IEEE International Conference on Industrial Informatics, Osaka, 2010, pp. 1085-1090.
9. R. Membarth, F. Hannig, J. Teich, M. Körner and W. Eckert, "Generating Device-specific GPU Code for Local Operators in Medical Imaging," 2012 IEEE 26th International Parallel and Distributed Processing Symposium, Shanghai, 2012, pp. 569-581.
10. Patlar Akbulut, F., Köseokur, H., Catal, C., Akbulut, A.: Son Kullanıcı Geliştirme için Otomatik Kod Üretim Aracının Tasarımı ve Gerçeklenmesi. Deu Muhendislik Fakultesi Fen ve Muhendislik. 19. 76-88.
11. T. D. Tran, Jie Liang and Chengjie Tu, "Lapped transform via time-domain pre- and post-filtering," in IEEE Transactions on Signal Processing, vol. 51, no. 6, pp. 1557-1571, June 2003.
12. K. Murota and K. Hirade, "GMSK Modulation for Digital Mobile Radio Telephony," in IEEE Transactions on Communications, vol. 29, no. 7, pp. 1044-1050, July 1981.
13. HDL Kodlayıcı Ana Sayfası, https://uk.mathworks.com/products/hdl-coder.html. Son erişim 09 Ekim 2018.
14. MATLAB Kodlayıcı Ana Sayfası, https://www.mathworks.com/products/matlab-coder.html Son erişim 09 Ekim 2018.
15. FFTW Kütüphanesi Ana Sayfası, http://www.fftw.org/. Son erişim 09 Ekim 2018.