Simulink HDL Kodlayıcı Tabanlı Paralel Mimari Uygulaması

Simulink HDL Coder Based Parallel Architecture Implementation

A. Esat Genç[1,2], Celal Güvendik[1,2], Aziz Gökgöz[1,2], Özgür Tamer[2], Metin Nil[1]

[1]Vestel Elektronik Ar-Ge
Vestel Elektronik San. ve Tic. A.Ş.
{ahmet.genc, celal.guvendik, aziz.gokgoz, metin.nil}@vestel.com.tr

[2]Elektrik ve Elektronik Mühendisliği Bölümü
Dokuz Eylül Üniversitesi
ozgur.tamer@deu.edu.tr

Özet

Görüntü işleme uygulamaları çarpma, toplama, çıkarma gibi bir çok işlemden oluşmaktadır. Bu işlemler ardışıl olarak yapıldığı için çok fazla saat periyoduna ihtiyaç duymaktadır. Bu işlemleri MATLAB Simulink ve HDL Coder kullanarak paralelleştirip işleyişi hızlandırmak ve sürecin tamamlanması için gereken saat periyodunu kısaltmak bu bildirinin odak noktasıdır. Paralel işlem yapabilmek için yüksek sayıda işlem elemenına ihtiyaç duyulmaktadır. Bununla birlikte yüksek sayıda eleman içeren paralel mimarilerin modellenmesi ve kurulması zaman alıcı ve zor bir işlem olduğu kadar hataya da açıktır. Bu tip mimarilerin görsel tasarımına izin veren araçların kullanımı, hataların belirlenmesini kolaylaştıracağı gibi geliştirme sürecini de hatırı sayılır oranda azaltacaktır. Bu metinde anlatılan çalışmada mimariler MATLAB Simulink görsel tasarım ortamında geliştirildikten sonra aynı yazılımın aracı olan HDL Coder kullanılarak donanım tanımlama dillerine çevrilmiştir. Bu bildiride, Matlab kodlarının Simulink HDL Coder kullanılarak gerek ön tanımlı bloklardan gerekse de uygulamaya has geliştirilmiş bloklardan donanım tanımlama dillerine çevrimi incelenmektedir

Abstract

Image processing applications contains many logic operations like product, summation, subrtaction etc. When an operation done by using iteration methode it requires many clock cycles. The focus of that artical is speeding up the processes and decrease the required clock cycles by using MATLAB Simulink and HDL Coder. For parallelization of the process many physical cores are needed. These huge number of cores brings some difficulties and easy to to make a mistake. MATLAB Simulink provides a visual design and helps to overcome possible design mistakes. In that paper the MATLAB Simulink is used to develop the desing and HDL Coder is used to translate the design to hardware language. In this paper, integration of Matlab script to Simulink HDL Coder, generating HDL code directly by using predefined Simulink blocks and generating a proper test bench using the HDL coder is investigated.

Giriş

FPGA teknolojisinin eşzamanlı birçok birimi aynı anda çalıştırabilme özelliği ve son yıllarda yaşadığı hızlı gelişme, bu yapıları yüksek başarım isteyen uygulamalar açısından popüler hale getirmiştir. Bağımsız paralel yapıların eşzamanlı programlanabilme özelliği sonucunda Sistolik Diziler gibi paralel mimarilerin modellenmesi açısından FPGA'ler uygun bir ortam sunmaktadırlar. Özellikle görüntü işleme uygulamalarında paralel çalışan işlemci sayısı yüzlere hatta binlere varabilmektedir. Bu durum sağladığı yüksek performans ve avantajlarının yanı sıra sistemin geliştirme ve uygulama prosedürlerinin çok daha karmaşık hale gelmesine neden olmaktadır. Çalışmamızın temelinde geliştirme aşamasında kullanılacak görsel bir geliştirme aracının tasarım karmaşıklığını azaltması ve olası hataların engellenmesi ve tespitinin kolaylaştırılması yatmaktadır.

Simulink HDL Coder kullanıcıların geliştirilen ve benzetimi yapılan model üzerinden sentezlenebilir donanım tanımlama kodları oluşturmasına olanak tanımaktadır. Bu sayede benzetim ortamı ile geliştirme ortamı eşlenebilmektedir.

Kod geliştirme için HDL Coder'a uygun Simulink blokları kullanılabileceği gibi, Matlab komut giriş satırı ya da Matlab kodları (.m dosyaları) da kullanılabilmektedir. Ek olarak benzetim ortamındaki şartları içeren bir test tezgahı (test bench) oluşturma yeteneği sunarak, yüksek seviye benzetim sonuçlarının doğrudan karşılaştırılabileceği bir mantıksal benzetim olanağı sunmaktadır [1].

Uyumluluk denetim programı oluşturulan donanım tanımlama kodu için modelin anlamını (semantic) incelemeyi sağlarken, HDL Coder'ın test tezgahı oluşturma yardımcısı oluşturulan test tezgahının doğruluk oranını artırmakta ve işlem zamanını kısaltmaktadır [1].

Paralel mimariler dijital görüntü işleme uygulamalarında sistemin çalışma süresini kısaltmak ve başarımını arttırmak için de kullanılmaktadır. Görüntü işleme uygulamalarında sıklıkla kullanılan kenar algılama işlemi, HDL Coder araç kutusu yapısının anlaşılmasını kolaylaştıracağı için bu çalışmada tercih edilmiştir.

2. Bölüm paralel mimarilerin tanıtımı ve gömülü sistemlerdeki kullanımıyla başlamaktadır. 3. bölümde Simulink HDL Coder tanıtılmış ve nasıl donanım tanımlama kodu oluşturulacağı anlatılmıştır. 4. Bölümde ise kenar algılama işlemi özetlenmiş ve örnek uygulama için sentezlenebilir donanım tanımlama

kodu oluşturulması tasvir edilmiştir. 5. bölümde de sonuçlar sunulmaktadır.

1. Paralel mimariler

Gerçek zamalı görüntü ve video işleme gibi birçok uygulamada yüksek kapasiteli işlem gücü gerekmektedir. Örnek olarak 512x512 pixel görüntünün kenar tespit (edge detection) işlemi için 2359296 çarpım, 262144 adet toplama ve bölme işlemi gerekmektedir.

512 x 512 pixellik bir görüntünün kenar tespit işlemini AMD Radeon E6760 gibi bir genel amaçlı bir işlemci 576 gflops ile 5µ saniyede yaparken Texas Instruments'in TMS320C6678 modelli özel dijital sinyal işleme kartı 160 gflops ile 18µ saniyede yapmaktadır. Yine aynı işlem FPGA'ler de 5.2µ [2] saniye gibi bir süre ile genel amaçlı işlemcilere benzer performans sergilemektedirler üstelik FPGA'in performansı paralel yapı kullanılarak attırılabilmektedir [3].

Paralelleştirme ile yapılan işler genellikle bir sistemin parçasını oluşturmaktadır. Görüntü işleme uygulamalarında kenar tespiti gibi birçok işlem gerçek zamanlı olarak belirli bir süre içerisinde gerçekleştirilmelidir. Gerçek zamanlı görüntü işleme konusunda birçok sıradan bilgisayar ve dijital sinyal işleme sistemleri başarısız olmaktadır. Fakat FPGA'ler içerisinde bir sürü çekirdek oluşturmaya ve eşzamanlı çalıştırmaya elverişli mimarileri ile bu sorunu aşma potansiyeli barındırmaktadırlar [4].

Eşzamanlı işleme mimarisi akıllı taşıtlar ve robotik görüş sistemleri gibi uygulamalarda yüksek performanslı görüntü ve video işleme fonksiyonları sağlamaktadır. Bu performans avantajına karşın eşzamanlı işleme elemanlarının sayısının artması ile sistem tasarımı ve testleri daha da karmaşıklaşmaktadır.

Düzgün bir paralel görüntü işleme fonksiyonu için çekirdek ayarları ya girdi olan görüntüye göre seçilmeli ya da girdi olacak görüntü türleri çekirdeğin yapısına uygun forma getirilmelidir. Bu konu görüntüyü alt parçalara ayırırken dikkate alınması gereken bir noktadır.

Sıradan bir algoritmayı paralel yapıya dönüştürürken iki tip problem ile karşılaşılmaktadır. Birincisi sıradan algoritmayı paralel yapıya dönüştürme ve girdileri, komutları eşzamanlı çalışacak şekilde düzenlemektir. İkinci problem ise zamanlamanın doğru yapılması ve parçacıklara ayrılıp işlenmiş girdinin doğru yerlerine geri konulmasıdır.

Dört çeşit paralel yapı mevcuttur [6];

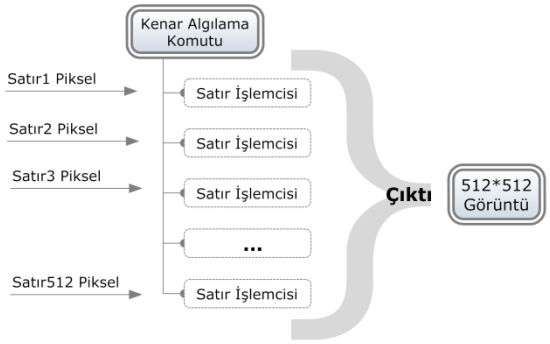
Bit seviyesinde paralelleştirme; bu yöntemde işlemcinin kelime uzunluğu ve elimizdeki verinin uzunluğu önem taşımaktadır. Kelime uzunluğu işlemcinin bir saat periyodunda (clock time) işleyebileceği maksimum veri uzunluğudur. Eğer işlenecek veri, işlemcinin kelime uzunluğundan daha fazla ise tamamlamak için birden fazla saat periyodu gerekecektir. Eğer veri uzunluğu işlemcinin kelime uzunluğundan kısa ise sadece bir saat periyodunda bu işlem yapılabilecektir. Örnek olarak 16-bit bir işlemci 10-bitlik bir toplama işlemini tek bir saat periyodunda gerçekleştirebilirken 8-bitlik bir işlemci bu iş için birden fazla saat periyoduna ihtiyaç duyacaktır.

Komut seviyesinde paralelleştirme; Programlar ve algoritmalar komut akışını kullanırlar. Bu komutlar tekrar düzenlenip veya birleştirilerek paralel yapıda çalıştırılabilecek şekilde gruplandırılabilir.

Veri paralelleştirmesi; En yaygın olarak kullanılan paralelleştirme yöntemidir. Bu başlık verinin aynı anda işlenebilmesi için işlem yapan birimler üzerine dağıtımına odaklanmıştır. Paralelleştirilen verinin ardışıl sistemlerdeki gibi birbirleri ile alakalı olması gerekmemektedir.

Paralelleştirmeye Michael J. Flynn tarafından sunulan Flynn'nin sınıflandırması tek komut birçok işlem yapısına örnek olarak verilebilir. Bu sınıflandırmada bir işlemci birçok veriyi tek bir komutla aynı anda işlemektedir.

Şekil 1 Bu çalışma sırasında Simulink'te oluşturulmuş tek komut çoklu veri yapısını, kenar algılama fonksiyonu için tanımlamaktadır.

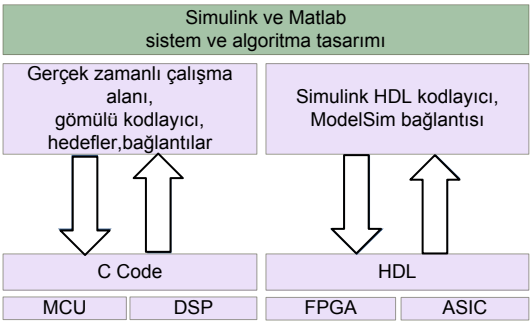


Şekil 1: Tek komut çoklu veri yapısı

Her ne kadar FPGA'lerin, aksiyonları paralel çalışan işlemciler üzerine dağıtma yeteneği olsa da giriş/çıkış bağlantıları, hat genişliği ve tampon hafızası gibi fiziksel sınırlamaları bulunmaktadır. Bu problemlerin aşılması için donanımın yenilenip geliştirilmesi gerekmektedir.

2. Simulink HDL kodlayıcı

Matlab Simulink® HDL Coder™, Simulink Model ve durum akış diyagramlarını temel alan sentezlenebilir donanım tanımlama dilini otomatik olarak çevirebilen bir araçtır. HDL Coder kullanarak, kullanıcılar Matlab ve Simulink'in içindeki sistem model ve algoritmaları tasarlayabilir, benzetimini yapabilir ve doğrulayabilirler. HDL coder otomatik olarak orijinal sistem ve algoritma modellerine karşı donanımsal ve yazılımsal uygulamalarını üretir ve doğrular. Günümüzde HDL Coder 2.1 versiyonunun desteklediği sentezlenebilir kod üretimi için ön tanımlı 160 adet blok bulunmaktadır [1].



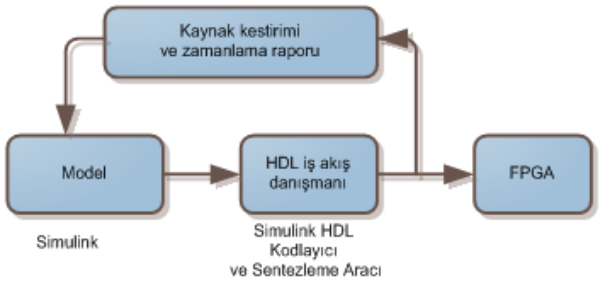
Şekil 2: Sistem ve algoritma tasarımı

Çok fazla bileşen içeren ve/veya paralel yapıdaki karmaşık sistemleri kurgulamak zor bir görevdir. Bu tür sistemleri geleneksel kod üretme teknikleri kullanılarak tasarımlamak, modüller arası yanlış bağlantılara neden olabilecek kadar karmaşıktır. Bu durumun önüne geçmek için, Simulink HDL Coder araç kutusu, sistem tasarımını kolaylaştırmak için

grafiksel kullanıcı arayüzü (GUI) sağlamaktadır. Simulink HDL Coderın önceden tanımlı blokları, Matlab komutlarını ve bütün FPGA tasarım sürecini destekleyerek, güvenilir ve güçlü yapıda fakat klasik yöntemlerle üstesinden gelinemeyecek kadar karmaşık sistem tasarımı sunmaktadır. Programlama dili yerine grafiksel kullanıcı arayüzü (GUI) kullanarak sistem bileşenlerini birleştirmek, kullanıcıya geliştirilen sistemi görsel olarak düzenleme olanağı sağlamaktadır ve tasarımı kolaylaştırmaktadır. Böylece, tasarım, doğrulama ve benzetim için harcanan süreler kısalmaktadır.

Bilindiği üzere, test tezgahı (test bench) tasarlanan modele giriş bilgileri vererek davranışını ölçer ve böylece sistemin doğruluğunu ve isabetini test etmiş olur. Fakat görüntü işleyen bloklar gibi çok fazla girişi olan sistemler için test tezgahı tasarlamak ve analiz etmek zor bir görevdir [3]. HDL Coder, Simulink koşulları altında ve tasarlanan modele bağlı otomatik test tezgahı üretim aracı içermektedir. Bu sayede, benzetim sonuçlarını elde etme süresi 2 haftaya kadar düşmektedir.

HDL Coder araç kutusu, diğer üretici firmaların araçlarında da kullanılabilen döngü içinde FPGA simülasyonlarını gerçekleştirebilen EDA Simülatör bağlantısıyla beraber tümleyici benzetim üretimini de desteklemektedir.



Şekil-3: FPGA dizayn iş akışı

Donanım tanımlama kodunu üretim işlemi, ilk olarak yapıyı Simulink'te tasarlamakla başlar. Tasarımda, blokların Matlab'ta kodlanması veya durum akışı veya Simulink HDL Coder kütüphanesinde ön tanımlı blokların kullanılması tercih edilebilir. Simulink'te geliştirilen sistemin parametreleri "HDL Coder Options" seçeneğinden düzenlenebilmektedir.

Bu çalışmada, model ön tanımlı bloklar yerine Matlab komutları kullanarak geliştirilmiştir.

"HDL Workflow Adviser", hedef cihazın ve uygun giriş/çıkış ara yüzlerin seçimini adım adım doğrulama şekliyle seçilmesine olanak sağlamaktadır. FPGA tasarım işleminde olduğu gibi bu araç kutusu, modelin uygunluğunu kontrol eder aynı zamanda HDL kodu ve test tezgahı üretim parametrelerini oluşturur, doğrular ve RTL kodlarının, RTL test tezgahının ve kosimülasyon modelinin birini veya hepsini üretir. Bu iş akışı, FPGA tasarım çevrim zamanını önemli ölçüde azaltır.

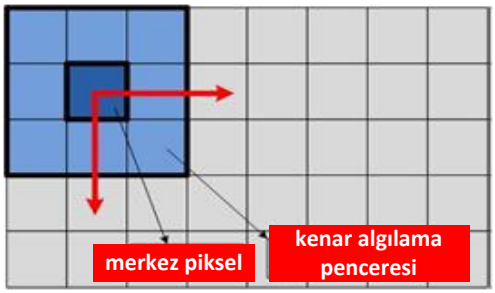
3. Uygulama

3.1. Kenar algılama

Kenar algılama işlemi imge bölütleme ve öznitelik çıkarma işlemlerinde kullanılan ön işlemlerden birisidir. Kenar algılama algoritmaları sayısal imgelerde bulunan dikey, yatay veya eğimli kenarları belirleyebilir.

Bu algoritmaları gradyan tabanlı ve sıfır-geçiş tabanlı olmak üzere ikiye ayırabiliriz. Gradyan tabanlı yöntemler uygulama yapılacak imgedeki en yüksek noktalar üzerinde işlem yaparken, sıfır geçiş tabanlı yöntemler ise imgenin ikinci dereceden türevi üzerinde çalışırlar [8].

Kenar algılama yöntemlerinde de algılama yapacak pencereler piksel piksel kaydırılarak işleme tabi tutulurlar. Bu pencereler imgenin üzerinde yatayda ve dikeyde kaymak suretiyle her adımda yapılan işlemler sonucu ilgili piksel için yeni değerler üretir. Bu aşamada penceredeki katsayılar üzerinde bulunduğu piksellerle çapılır ve elde edilen değerlerin toplamı pencerenin merkezindeki piksele atanır (Şekil 4).



Şekil 4: 3x3 boyutundaki pencere ile kenar algılama

Gradyan temelli yöntemler kenar algılamak için birinci dereceden türev kullanır. Sobel, Prewitt ve Roberts (Tablo 1) gredient tabanlı bu yöntemlerde kullanılan pencerelerden bazılarıdır [9]. Keskin hatlara sahip kenarları olan imgelerde birinci dereceden türev kullanan bu yöntemler daha başarılı sonuçlar verir [3]. Çünkü imgelerdeki ögelerin sınırlarındaki renk tonu geçişleri ne kadar büyük olursa birinci dereceden türev ilgili pikselde en büyük değerine ulaşacaktır ve kenar saptanacaktır.

Tablo 1: 3x3 dikey kenar (sol) ve yatay kenar (sağ) algılayıcı Sobel pencereleri.

-1	0	1
-2	0	2
-1	0	1

-1	-2	-1
0	0	0
1	2	1

Sıfır geçiş temelli yöntemler kenar geçişleri daha yumuşak olan imgeler için kullanılabilir. Bu yöntem ise imgenin ikinci türevindeki sıfır geçiş noktalarını arar. Bu noktalar ise renk tonu değişimlerinden dolayı kenarlarda oluşur. Bu nedenle daha yumuşak renk tonu geçişlerine sahip imgeler için zero-crossing yöntemi gradient yönteminden daha iyi sonuç verir [3]. Tablo 2 de gösterilen Lablacian penceresi zero-crossing tabanlı yöntem için kullanılabilir.

Tablo 2: 3x3 Laplacian penceresi

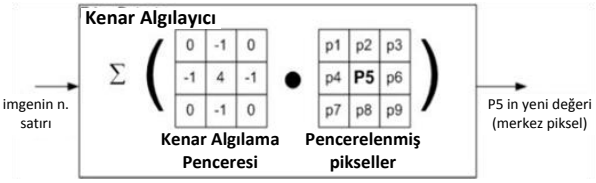
0	-1	0
-1	4	-1
0	-1	0

Ayrıca bahsi geçen bu kenar algılama pencereleri birbirleriyle evrişime tabi tutularak yeni pencereler elde edilebilir. Böylece pencerelerin başarımı veya işlevi arttırılabilir.

3.2. Simulink'te paralelleştirilmiş gömülü kenar algılama aracı ve HDL kod üretimi

3.2.1. Kenar algılama aracı bütünleştirmesi

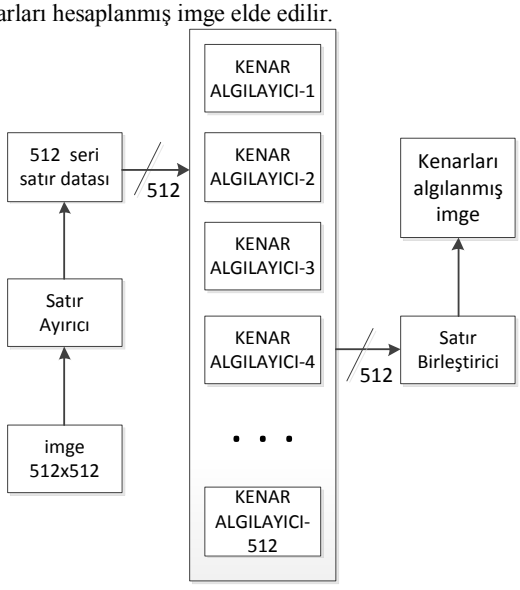
Simulinkte önceden tanımlanmış bazı kenar algılama blokları mevcuttur. Fakat bu çalışmada kullanılan kenar algılama blokları, Matlab kodları ve Simulink araçları kullanarak, Simulink HDL kodlayıcıya uygun bir biçimde, paralel bir mimariyle tasarlanmıştır.
Algoritma, kenarları Şekil 5'te gösterilen Laplacian penceresi kullanarak algılamaktadır. P5 pikseli kontrol edilerek bir kenara ait olup olmadığına karar verilir.



Şekil 5: p1, p2, p3 imgenin (n-1). satırında , p4, p5, p6 imgenin n. satırında and p7, p8, p9 imgenin (n+1). satırındadır.

3.2.2. Paralelleştirme

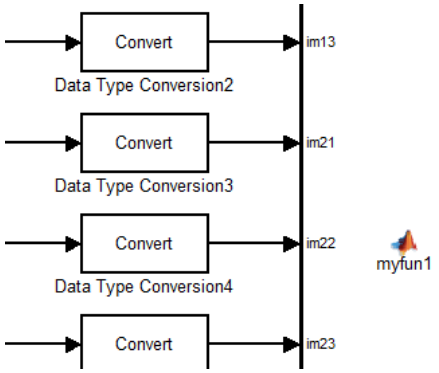
Kenar algılama algoritmasını paralel işlemler haline dönüştürmek için her bir piksel satırını bir küçük işlemcide işleyip, satır sayısı oranı kadar işlem süresinden kazanç sağlanması hedeflenmektedir. Böylece her küçük işlemci sadece kendi satırında işlem yapacaktır. Şekil 6 da paralelleştirilmiş yapının şematik gösterimi verilmiştir.
İlk blok 512x512 lik sayısal bir resmi satırlarına ayırmaktadır. Sonraki aşamada bu satırlar yeni paket veriler olarak kabul edilip seri hale getirilmektedir. Böylece 512 tane piksel bilgisini içeren 512 tane satır, tasarlanan kenar algılama aracı işlemcilere seri olarak gönderilmektedir. Bu araç merkez pikselin değerini hesaplamakla görevlidir. Bu paralel bloklardan sonra hesaplanan değerler tekrar seri akış olarak sonraki bloğa aktarılır. Burada satırlar tekrar birleştirilerek kenarları hesaplanmış imge elde edilir.



Şekil 6: Paralel kenar algılama aracı şematik gösterimi

3.2.3. HDL kod üretimi

Simulink HDL kodlayıcı uygun blok tasarımları yapıldığı takdirde HDL kodu üretebilmektedir. Bu çalışmadaki paralel kenar algılayıcı Simulink HDL kodlayıcıya uygun bir yapıda tasarlanmış ve VHDL kodlar hedef FPGA için üretilmiştir. Tasarımların gerçek donanıma uygun olması Simulink HDL kodlayıcıda sentezlenebilmesi için önemlidir.
Bu çalışmada da gerçek donanıma uygun bir yapı olmasına özen gösterilerek Simulink HDL kodlayıcıya VHDL kodları sentezlendirilmiştir. Örneğin; Simulink HDL kodlayıcı matris formatındaki verilere herhangi bir bloğun girişinde veya çıkışında izin vermemektedir. Bu sebeble çalışmadaki giriş ve çıkışlar seri veri akışı şeklinde düşünülmüştür. Ayrıca tüm girişler fixed-point veri tipine çevrilerek gerçek donanıma uygunluk sağlanmıştır (Şekil 7). Fixed-point data tipi sistem donanımına uygun yuvarlanmış sayılardır.



Şekil 7: Simulink HDL Kodlayıcıda Fixed-Point data tipi çevirimi ve seri data akışları

4. Sonuçlar

Bu çalışmanın sonuçları 2 şekilde analiz edilebilir; tasarımın işlem süresi ve donanım performansı. Bildiride yapılan çalışmada kenar algımala algoritması paralelleştirilerek 512x512 piksel boyutundaki bir görüntüye uygulanmıştır. Kullanılan paralel yapı, aynı komutla bir bilgi dizisini yöneten SIMD (Single Instruction Multiple Data) mimarisini temel almaktadır.
Çalışmada, paralel kenar belirleme algılama algoritması 512x512 boyutundaki görüntüye uygulandı. Kullanılan paralel yapı, aynı komutla bir bilgi dizisini yöneten SIMD mimarisini temel almaktadır. Alınan görüntüden çoklu görüntü elde etmek için diziler kenar algılama işleminden önce bölünmüş ve seri hale getirilmiştir. Her dizi algoritma tarafından alınmış, toplanmış ve ardından geliştirilen kenar algılama metodu uygulanmıştır. Bu sayede performans artmış ve işlem süresi kısalmıştır.
Simulink HDL Coder, HDL kod yazmaya nazaran modelin geliştirme ve uygun test tezgâhı oluşturma hızını da artırmaktadır.
Sentezlenebilir kod ve test tezgâhı ile beraber modelin geliştirilmesi bu çalışma için yaklaşık 40 adam/ay sürmüştür. Aynı tasarımın HDL kodları kullanılarak geliştirilmesi genel olarak 550 adam/ay sürdüğü göz önünde bulundurulursa, geliştirme süresinin çok kısa olduğu görülebilmektedir [10].
Paralel yapılar gibi elemanlar içeren çoklu işlemci mimarisi gibi karmaşık sistemler için donanım tasarımı ve onayı bu bildiride öngörüldüğü gibi Simulink HDL Coder kullanılarak kısa sürede gerçeklenebilir.

Gelecek çalışmalar için, tasarlanan yapı FPGA'de gerçeklenecek ve gerçek zaman değerleri ölçülecektir.

5. Kaynaklar

[1] Matlab, (2011), *Simulink HDL Coder 2.1* [Online]. Mevcut: http://www.mathworks.de/products/datasheets/pdf/simulink-hdl-coder.pdf

[2] P. Mandl, U. Bordoloi "General-purpose Graphics Processing Units Deliver New Capabilities to the Embedded Market" Boards and Solution Magazine, Ekim 2011.

[3] Ö.Tamer, L. A. Rønningen, M. Panggabean, "Real Time Edge Detection Using Three Dimensional Systolic Array", *Parallel and Distributed Computing and Systems*, USA,8-10 Kasım 2010.

[4] A. Cosoroaba, F. Rivoallon, (2006, Temmuz 5), *Achieving Higher System Performance with the Virtex-5 Family of FPGAs* [Online]. Mevcut: http://www.xilinx.com/support/documentation/white_papers/wp245.pdf

[5] Y. Li, Q. Yao, B. Tian, W. Xu, "Fast Double-Parallel image processing based on FPGA", *IEEE International Conference on Vehicular Electronics and Safety*, Beijing, China, 10-12 Temmuz 2011.

[6] D. A. Patterson and J. L. Hennessey, "Computer Organization and Design: the Hardware/Software Interface", 2. Baskı, Morgan Kaufmann Publishers, San Francisco, California, 1998, s.751.

[7] Culler, David E., Singh J.P., and Gupta A., Parallel Computer Architecture - A Hardware/Software Approach, 1. Baskı, Morgan Kaufmann Publishers, San Francisco, 1998, s. 15.

[8] E. Nadernejad, S. Sharifzadeh, H. Hassanpour, "Edge Detection Techniques: Evaluations and Comparisons Applied Mathematical Sciences", Cilt 2, no. 31, 2008, 1507 – 1520.

[9] H. S. Neoh, A. Hazanchuk, "Adaptive Edge Detection for Real-Time Video Processing using FPGAs", Global Signal Processing Expo and Conference, Santa Clara, USA, 2004.

[10] MediaTek Inc. (2009, Eylül), *Automatic Hardware Implementation of Digital Filters for an Audio Codec* [Online].Mevcut: http://www.mathworks.com/company/newsletters/digest/2009/sept/audio-codec.html