

Sayısal Hücresel Sinir Ağı Benzetimi CASTLE Mimarisinin Temelleri

(Digital Cellular Neural Network
Simulation and Fundamentals of
CASTLE Architecture)

Erhan Küçükğüzel
2005

İçindekiler :

<u>1. Giriş.....</u>	<u>3</u>
<u>2. Sistem özellikleri</u>	<u>3</u>
<u>3. Matematik Model.....</u>	<u>3</u>
<u>4. İşlemci Birimi Mimarisi ve Yöntem.....</u>	<u>4</u>
<u>Sonuç ve Yorum.....</u>	<u>8</u>
<u>Kaynakça :.....</u>	<u>9</u>

1. Giriş

Bu raporda, ilk kez 1998 yılında Londra’da gerçekleşen 5.IEEE Uluslar arası CNNA Konferansında sunulan bildirideki [1], HSA nın sayısal bir benzetimi olan CASTLE adlı işlemci mimarisi incelenmiştir. İncelemede, bu mimarinin yapısı ve işlevi üzerinde durulmuştur.

CASTLE Mimarisinin, rapora konu olmasındaki gerekçe, sayısal HSA benzetiminde popüler bir yapı oluşudur. Aynı konuda birden fazla yayına konu olmuş bir çalışma olduğu gibi, alternatif yöntemler, başarımlar karşılaştırmalarında bu mimariye atıfta bulunmuşlardır. Çalışmamızda bize yol gösterici olacağına inanıyorum.

2. Sistem özellikleri

[1] in sonuç bölümünde anlatılanlardan, CASTLE mimarisinin özel tasarım sayısal bir tümdevrede (ASIC) olarak gerçekleştirildiği anlaşılmaktadır. Sistem mimarisini geliştirebilir sistemler sınıfına sokmaktadırlar (evolvable system). Konuyla ilgilenen “Analogic and Neural Computing Systems Laboratory” ortamında özel tasarım ortamı oluşu bunun sebebi olabilir [3]. Tamamen sayısal olan bu yapının, istenirse FPGA üzerinde gerçekleştirilebileceğini, konunun uzmanlarına sorarak öğrenmiş bulunuyoruz. Ayrıca, bu mimariye rakip, FPGA üzerinde gerçekleştirilmiş çalışmalara literatürde rastlanmaktadır [4,5].

[1] bildirisi ile sunulan yapı, [2] makalesi ile detaylı olarak anlatılmıştır. Başarıma ait bilgi şöyledir :

Hız : 1ns / sanal hücre / HSA iterasyon Çözünürlük : 12 bit

Bu sayede, saniyede 25 adet 240x320 pikseli resim, her pikseli için 500 iterasyon yapılmak üzere filtrelenilmektedir. Bu hız NTSC standardındadır.

Verilen sayıları inceleyelim :

25 x 240 x 320 x 500 = 960 000 000 = saniyedeki 1 hücrelik hesap sayısı

Hesap başına düşen süre 1 ns civarında olmaktadır. Aslında gerçek hesap süresi daha uzun olsa da paralellikten dolayı piksel başına süre düşük görünüyor.

CASTLE yapısı tamamen paralel bir yapı olmamakla birlikte, tamamen seri bir yapı da değildir, belli derecede paralellik taşır. Resmin tüm pikselleri tümdevre içine alınıp aynı anda işlenmez, bunun yerine seri olarak alınır ve satır satır (veya sütun sütun) işlenir. Bu yapının tercih edilmesinin bir sebebi, orta büyüklükteki bir resmin her pikseli için tek tümdevre içinde yeterli sayıda işlemci kurmak çok zordur. Gene de mümkün mertebe paralel işleme sonucu hız, tek bir mikroişlemci ve yazılım çözümüne göre açık ara üstündür.

3. Matematik Model

HSA hesapları için Chua-Yang modelinden yola çıkılmıştır :

$$\dot{x}_{ij}(t) = -x_{ij} + \sum_{C(kl) \in N_r(i,j)} A_{ij,kl} y_{kl}(t) + \sum_{C(kl) \in N_r(i,j)} B_{ij,kl} u_{kl}(t) + z_{ij} \quad (1)$$

İleri Euler formülü ile türev alınırsa :

$$\frac{x_{ij}(n+1) - x_{ij}(n)}{h} = -x_{ij}(n) + \sum_{C(kl) \in N_r(i,j)} A_{ij,kl} y_{kl}(t) + \sum_{C(kl) \in N_r(i,j)} B_{ij,kl} u_{kl}(t) + z_{ij} \quad (2)$$

$$x_{ij}(n+1) = (1-h)x_{ij}(n) + h \left(\sum_{C(kl) \in N_r(i,j)} A_{ij,kl} x_{kl}(t) + \sum_{C(kl) \in N_r(i,j)} B_{ij,kl} u_{kl}(t) + z_{ij} \right) \quad (3)$$

Bu adımda kırpm (truncation) işlemi gerçekleştirilerek, x ve y terimleri birleştirilmiştir.

$$\hat{A} = h \begin{bmatrix} a_{-1,-1} & a_{-1,0} & a_{-1,1} \\ a_{0,-1} & \frac{1+h(a_{0,0}-1)}{h} & a_{0,1} \\ a_{1,-1} & a_{1,0} & a_{1,1} \end{bmatrix}, \quad \hat{B} = h.B \quad (4)$$

A ve B matrisleri yukarıdaki şekilde düzenlenirse Chua-Yang modelinden FSR (Full Signal Range) modeline aşağıdaki gibi geçilebilir :

$$x_{ij}(n+1) = \sum_{C(kl) \in N_r(i,j)} \hat{A}_{ij,kl} x_{kl}(n) + g_{ij} \quad (5)$$

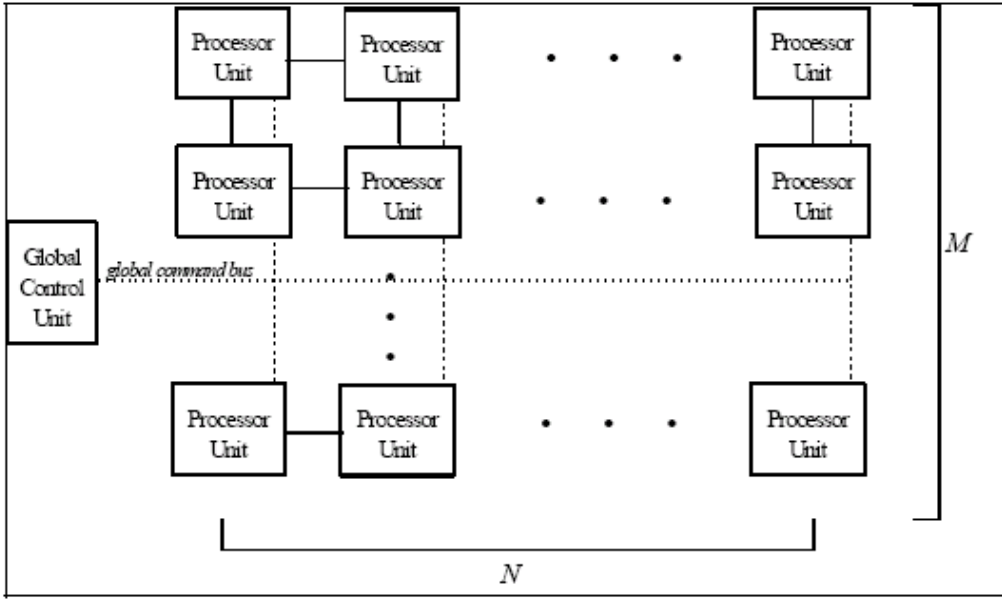
$$g_{ij} = \sum_{C(kl) \in N_r(i,j)} \hat{B}_{ij,kl} u_{kl}(n) + h z_{ij}$$

Tümdevrede işlemler bu modele dayanmaktadır. Bu modelin seçilmesindeki sebepler şunlar olabilir :

- 1- g_{ij} her piksel için bir kez hesaplanır, yani bu modelle her iterasyon için değişen ve sabit kalan kısımlar ayrılmış olur
- 2- Her iki denklem de iki tane 3x3 matrisin noktasal çarpımı ve bir sabit toplamından oluşur, yani aynı işlemci yapısıyla sadece parametreleri değiştirerek her iki denklem de hesaplanabilir.

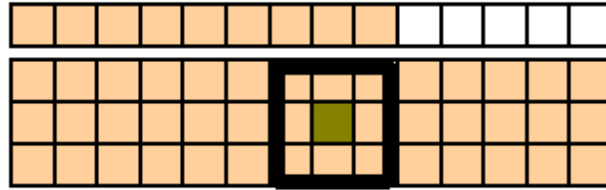
4. İşlemci Birimi Mimarisi ve Yöntem

Resim tümdevreye alınırken, 240 piksel genişlik 6 parçaya bölünerek 6 adet 40 piksel genişlikte ince uzun şerit elde ediliyor. Şekil 1 deki işlemci birimlerinden her sütun bir şerite atanmıştır. Ana kumanda birimi (Global Control Unit) tüm işlemcilere yapılacak işle ilgili komutları yollamak ve senkronizasyonu sağlamakla yükümlüdür. Komutlar ana kumanda yolu (global command bus) üzerinden (Şek.1, Şek.3) gönderilir. Tasarımda birçok saat işaretinin doğru zamanda oluşturulması gerekmektedir. Üretilen saat işaretleri, işlemcide kaydırmaların, yazmaçlara bilgi yüklemenin, işlemlerin doğru sırada ve tekrarda yapılmasını sağlar. Bu işaretlerin üretilmesi, lojik devrelerde “Sonlu Durumlu Makine” (FSM : Finite State Machine) konusuna girer. İşlemcideki zamanlama ve kontrol birimi (Şek.3) bu işe bakar.



Şekil 1 : Tümdevredeki işlemcilerin dizilişi

Her işlemci birimine pikseller seri olarak gelmektedir. İşlemciye tahsis edilmiş piksel şeridi 3 satır olup, ayrıca işlemler sürerken yüklemesi devam eden bir satırı (hücre belleği) daha vardır (Şekil 2).

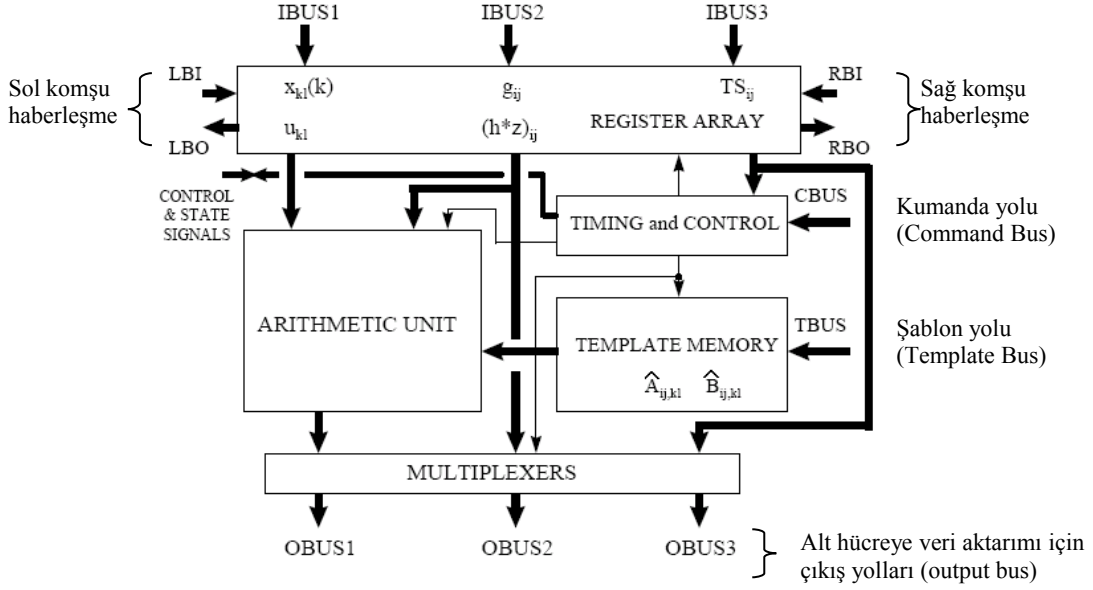


Şekil 2 : Tek işlemciye ait piksellerin alınışı ve işlenişi

Şekil 2 de bir işlemciye ayrılmış resim parçasının sembolik gösterimi vardır. Yukarıdan aşağı satırları 1,2,3,4 olarak numaralandırırsak, koyu çerçeve içindeki 3x3 genişlikte kısım farklı pikselleri dolaşırken, her defasında 3. satırın pikselleri için durum (çıkış) hesabına ait tek bir iterasyon yapılmaktadır. Bu satırın hesabı sonunda 1. satıra gelmiş olan piksellerin (durumu ya da girişi) alınışı de tamalanmış olup, tüm pikseller dikey olarak 1 satır aşağı kaymaktadır. Böylece bir üst satırın çıkış hesabına geçilebilir. Elde edilen sonuçlar da bir sonraki işlemciye, o işlemcinin ilk satırına yazılarak aktarılır. Aktarılan satır orada bir sonraki iterasyona katılmak üzere sırasını bekler.

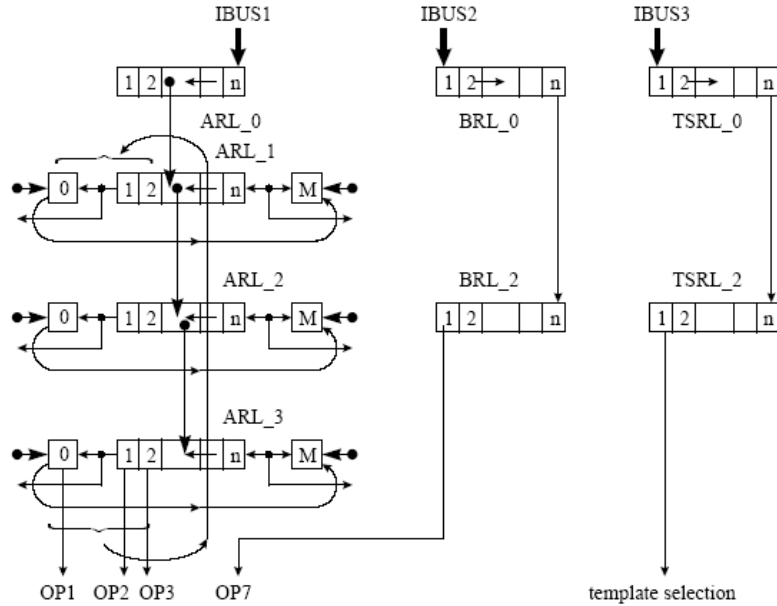
İşlemcide 3x3 çarpma ve toplama işlemi için kurulu olan yapı, önce (muhtemelen ilk işlemci satırında) g_{ij} değerlerini hesaplamak için kullanılır. Daha sonra bu değerler diğer işlemcilere dikey yolla iletilir ve işlemlere sabit olarak eklenir.

Resme ve filtreye ait girişlerimiz; u_{ij} resim bilgisi, $h * z_{ij}$ sabitleri, \hat{A} ve \hat{B} şablonları
Çıkışlar (iterasyonların girişleri) : x_{ij} ler
Tüm bu bilgiler tümdevreye ilgili veriyolları üzerinden girilir.



Şekil 3 : İşlemci blok diyagramı

Giriş yolu IBUS1 u_{ij} , x_{ij} değerlerini taşır. Buradan anlaşılabilir, ilk aşamada u_{ij} girilir. İlk işlemciler g_{ij} değerlerini hesaplar, daha sonra bu yol x_{ij} durumlarını taşır. IBUS2 de, görünüme bakılırsa, ilk satırda dışarıdan verilen $h*z_{ij}$ değerlerini işlemciye taşıyarak g_{ij} hesaplarına katkıda bulunur. Daha sonra bu yol, sabit eklenti olarak g_{ij} leri taşıyacaktır. IBUS3 ise şablon seçim vektörlerini iletir. Şekil 3 teki şablon belleği, daha önceden şablon yolundan gönderilen 16 adet şablonu saklar. Şablon seçim vektörü ise işlemcilere, kullanılacak şablonu, her piksel için farklı olabilecek biçimde iletir. Bu özellik, [2] de örneği verilen Pusula Filtre (Compass Filter) gibi birden fazla şablon kullanılan algoritmaların da tümdevre yardımıyla uygulanabilmesini sağlar.

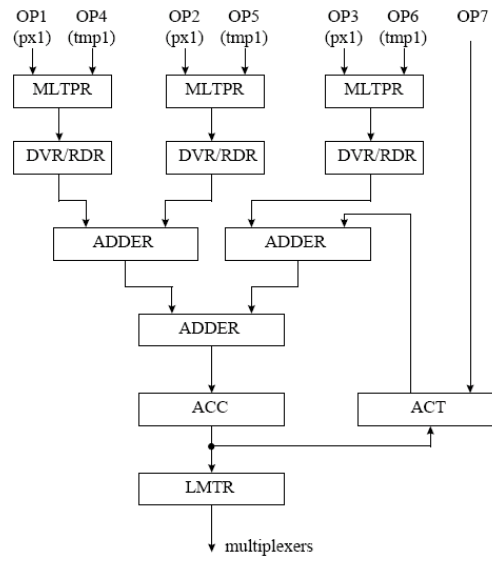


Şekil 4 : Yazmaç dizileri

RBI, RBO, LBI, LBO yolları yanal komşu işlemcilerle haberleşmeyi sağlar. İşlemcinin payına düşen en sağ ve en sol piksellerin hesabında diğer işlemciye ait piksel değeri gerektiği için bu yapı kullanılır. İşlemci, komşuya ait pikselleri, şekil 4 te görülen “0” ve “M” kodlu yazmaçlarda saklar. İşlemci en sağ veya sol kenarda ise bir tarafında komşu olmayacağından bu yazmaçlar 0 yüklü olacaktır.

OBUS1 çıkış yoludur ve piksellere ait hesaplanan çıkışları (x_{ij}) alt komşu işlemciye iletir. Büyük ihtimalle ilk işlemci satırının çıktıları g_{ij} değerleridir. OBUS2 de piksellere ait sabitleri (g_{ij}) alt komşu işlemciye taşır. Gene muhtemelen ilk defasında $h * z_{ij}$ taşınır. OBUS3 de şablon seçim vektörünü sürekli alt komşulara iletir. Tüm bu giriş ve çıkışlar eşzamanlı olarak, en son işlemciye kadar birlikte seyahat etmek zorundadır.

İşlemciye ait aritmetik işlem biriminin yapısı Şekil 5 te görülmektedir. 3x3 nokta çarpım için 9 adet çarpımı birbiriyle ve sabit terimle toplamak gerekmektedir. Hesap yapılacak 3x3 piksellik bölge Şekil 4 te görülen biçimde sola kaydırılırken, her hesap için 3 alt adım vardır. Bu adımlarda 3x3 piksellik kısımlar 3 kez dikey olarak aşağı kaydırılarak aritmetik işlem biriminin girişleri OP1, OP2, OP3 elde edilir (en alt 3lü satır en üste kayar, böylece 3 adım sonunda 3x3 piksellik seçilen kısım ilk haline gelmiş olur). Bu operandlar şablon matrisinden gelen (şablon belleğinde saklı) değerlerle çarpılır, bir önceki 3lü çarpımın sonucu veya sabit terime eklenir. 3 adım sonunda 9 çarpım sona erer ve LMTR etiketli Kırpıcı ile doğrusal olmayan fonksiyondan geçerek çıkış elde edilir.



Şekil 5

Çarpıcılar tümdevre tasarımında alan kaplayan yapılardır. Öte yandan aynı anda ne kadar çok çarpma yapabilirsek o kadar hız kazanırız.

Toplama işlemleri için (Şek.5 ADDER) tam toplayıcı kullanılır (Şek.6). En düşük değerli bitin işlemine ait elde, bir üst değerli bitin toplamına aktarılarak toplama işlemi devam eder. Bit sayısı arttıkça eldenin son bite ulaşması, işlemleri yavaşlatacak derecede uzun sürer. Buna karşı, CASTLE yapısında da olduğu gibi [1] İleri bakmalı toplayıcılar (LACA=Look Ahead Carry Adder) kullanılır [6]. Bu yapıyı hatırlamak için tam toplayıcının (Şek.6) matematik modelini yazalım :

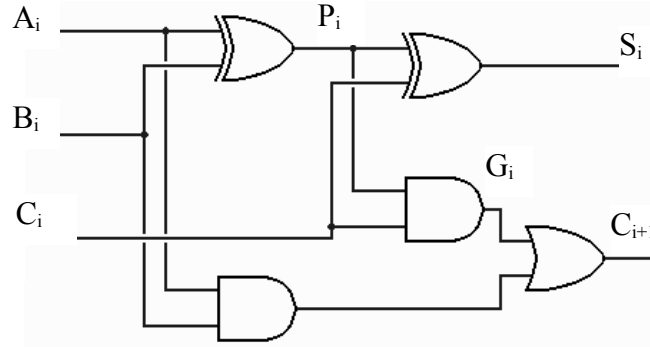
$$P_i = A_i \oplus B_i, \quad G_i = A_i \cdot B_i, \quad S_i = P_i \oplus C_i, \quad C_{i+1} = G_i + P_i \cdot C_i \quad (6)$$

$$C_2 = G_1 + P_1 C_1 \quad (7)$$

$$C_3 = G_2 + P_2 C_2 = G_2 + P_2 (G_1 + P_1 C_1) = G_2 + P_2 G_1 + P_2 P_1 C_1 \quad (8)$$

$$C_4 = G_3 + P_3 C_3 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 C_1 \quad (9)$$

Bu ifadelerde eldelerden sadece C_1 var, P_i ler ise paralel olarak aynı anda üretildiği için eldelerin hesaplanması 2 kapı gecikmesi süresinde tamamlanmaktadır. Tabi bu kez yukarıdaki ifadeleri veren devrenin gerçekleşmesi gerekmektedir.



Şekil 6 : Tam toplayıcı

Sonuç ve Yorum

CNN-UM nin mucidi olan ekibin gerçekleştirdiği bu sayısal HSA benzetimi, analog çözüme göre daha yavaş da kalsa, doğrulukta ondan ileridedir. Maliyetinin çok daha düşük oluşu ve hareketli resimlerde bile yeterince hızlı oluşu çalışmaların sayısal yapıya kayacağını düşündürmektedir. Bu yapıda daha sonradan yapılan geliştirmeler ile hız artırılmıştır [7] ancak raporda bu gelişmelere yer verilmemiştir. Aynı yapı daha gelişmiş bir tümdevre ile daha fazla hücreyi aynı anda işleyerek daha yüksek hızlara çıkabilir. İlgili yayınlarda, resim bilgilerinin son işlemciye ulaşmasından sonra nasıl iterasyona devam ettiği anlatılmamıştır (bahsedilen 500 iterasyon için yeterince işlemci bulunmamaktadır). Bu konuda yapılan muhtemelen çıkışların bir bellekte tutularak, tüm piksel girişi bittikten sonra tekrar tümdevreye girmesi şeklinde olabilir. CASTLE yapısının bir üstünlüğü de (karmaşıklığını artıran bir sebep) çözünürlüğünün ayarlanabilir olmasıdır. Tek bit çözünürlük ve yüksek çözünürlükler için aynı işlemler yapılmaz. Böylece düşük çözünürlükte hız da artar. Görüldüğü gibi birçok amaca hizmet edebilen bir yapı olarak karmaşıklığı da yüksektir. Amaç daraltıldığı takdirde daha basit yapılara gidilebilir. Çalışmamızda küçük yapı taşlarından hareketle tümevarım hedeflenmelidir. Başarılı bulduğum CASTLE yapısının, iyi bir ekip çalışmasından doğduğu anlaşılmaktadır.

Kaynakça :

- [1] Zarandy A., Keresztes P., Roska T., Szolkay P., “An emulated digital architecture implementing the CNN Universal Machine”, proc. of the fifth IEEE Int. Workshop on Cellular Neural Networks and their applications, London pp: 249-252, April, 1998.
- [2] Keresztes P., Zarandy A., Roska T., Szolkay P., Bezak T., Hidvegi T., Jonas P., Katona A., “An emulated digital CNN implementation”, Journal of VLSI Signal Processing Systems Kluwer Academic Publishers, Vol. 23. pp. 291-303, 1999.
- [3] İnternet sitesi, “Analogic and Neural Computing Laboratory”, Computer and Automation Research Institute of the Hungarian Academy of Sciences, <http://lab.analogic.sztaki.hu/>
- [4] Perko M., Fajfar I., Tuma T., Puhan J., “Low-Cost, High-Performance CNN Simulator Implemented in FPGA”, 6th IEEE International Workshop on CNN and their applications, 2000.
- [5] Nagy Z., Szolgay P., “Configurable Multi-Layer CNN Emulator on FPGA”, Proc. of IEEE CNNA02, World Scientific, pp. 164-171, Frankfurt, 2002.
- [6] Eriş, E., “Eldenin Yayılması”, Lojik Devreler ders notları, say. 57-59.
- [7] Hidvegi, T., Keresztes P., Szolgay P., “An Accelerated Digital CNN-UM (CASTLE) Architecture Using the Pipe-line Technique”, World Scientific, April 2002.