6. BÖLÜM

PVC TİPİ ARİTMİ SINIFLANDIRICININ FPGA DONANIMININ OLUŞTURULMASI

Programlanabilir Lojik (Programmable Logic), temel lojik fonksiyonları kullanarak kompleks donanım modelleri ortaya koymaya yarayan bir donanım geliştirme platformudur. Evdeki yüksek çözünürlüklü televizyondan, yakınlardaki GSM vericisine, bankadaki ATM makinesinden, araçtaki navigasyon cihazına kadar çok çeşitli alanlarda programlanabilir lojik kullanılmaktadır. Modern Kompleks Programlanabilir Lojik Devreler (CPLD, Complex Programmable Logic Devices) ve Alan Programlanabilir Kapı Dizileri'ne (FPGA, Field Programmable Gate Arrays) kadar gelişen programlanabilir lojik teknolojisi, Texas Instruments'in 7400 serisi entegre devre elemanlarını bir Baskılı Devre Kartı (PCB, Printed Circuit Board) üzerinde birbirlerine bağlamakla başlar. 1960 ve 1970'li yıllarda birçok tasarım, içerisinde temel lojik kapılar, flip floplar ve bazı özel lojik yapıların bulunduğu 7400 serisi TTL (Transistör Transistör Lojik) entegre devre elemanları ile yapılmıştır [119].

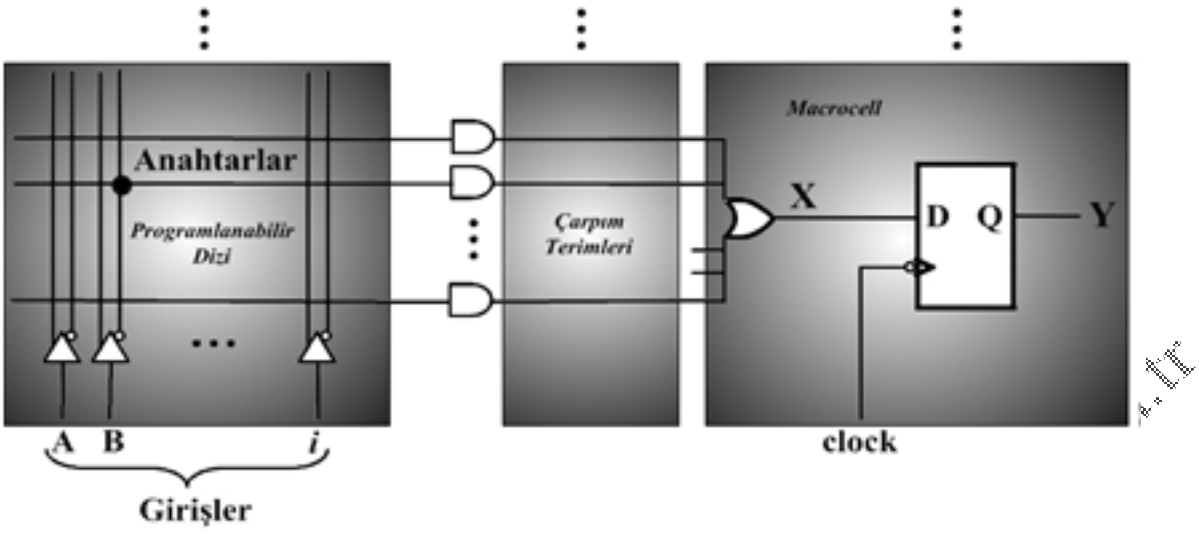
TTL ile tasarımda amaç genellikle, mümkün olduğunca az sayıda entegre devre kullanarak maliyeti düşürmek ve devre kartının alanını küçültmektir. Devre kartı yapılmadan önce, tasarımcının devreyi kafasında planlamış olması gerekmektedir. Örneğin bir tasarımda ihtiyaç duyulan bir OR kapısı için fazladan entegre devre kullanmak yerine, var olan NAND kapılarını kullanarak daha az sayıda devre elemanı kullanılmış olacaktır. Bu ve benzeri optimizasyonların, lojik fonksiyonun cevabını değiştirmediğinden emin olmak için devre gerçeklemesine geçilmeden önce tasarımın doğrulanması gerekmektedir. Doğruluk tablosu ve Karnough haritası çıkarılarak fonksiyon cevabı üretilmiş olan ifade, gerektiğinde Boolen denklemlerine başvurularak en az sayıda devre elemanı ile veya tasarımcının elinde bulunan devre elemanlarına göre son halini alır.

Birçok fonksiyon çarpımların toplamı şeklinde ifade edilebilir veya sadeleştirilebilir. Bu tür fonksiyonlar iki lojik seviyede, AND kapıları ile çarpımlar ve OR kapıları ile toplamlar olmak üzere, çarpımların toplamı fonksiyonu şeklinde gerçekleştirilebilir. TTL lojik devreler ile yapılan tasarımlarda, bu fonksiyonları icra eden entegre devrelerin devre kartı üzerinde birbirlerine bağlanması gerekmekteydi. Programlanabilir Lojik Devreler (PLD, Programmable Logic Devices) ile bu fonksiyonlar genelleştirilerek tek bir entegre devre şeklinde üretilmiştir.

Programlanabilir Lojik Diziler (PLA, Programmable Logic Arrays) ilk PLD entegre devreleridir. PLA entegre devrelerinde hafıza elemanı bulunmamaktadır. Çıkışların birbirleri ile senkronizasyonu veya çıkış değerlerinin saklanması gerekmiyorsa, hafıza elemanının kullanılmasına gerek yoktur. Ancak çıkışın diğer çıkışlar ile senkronize olması veya saklanması isteniyorsa, bu durumda çıkışlar için birer flip flop yani hafıza elemanı gerekir. Şekil 6.1'de verilen şematik diyagramda, X çıkış değerine kadar olan kısım, iki lojik seviyeden oluşan PLA yapısını göstermektedir [120].

Advanced Micro Devices firması tarafından üretilen Programlanabilir Dizi Lojik (PAL, Programmable Array Logic) entegre devrelerinde, çarpımların toplamları fonksiyonu çıkışı (X), bir hafıza elemanına giriş olarak uygulanmıştır. Bu şekilde PLA mimarisi geliştirilmiş ve PLD entegre devrelerine senkronizasyon ve veri saklama özellikleri kazandırılmıştır. Çarpımların toplamı çıkışı ile hafıza elemanının birleştirildiği bu yapıya Macrocell denir. PLA'nın hafıza elemanı eklenerek geliştirilmesi sonucu ortaya konulan temel PAL mimarisi Şekil 6.1'de verilmiştir. Burada X kombinasyonel (combinational), Y ise sıralı (sequential) PAL çıkışları olarak ifade edilebilir. Şekil 6.1'de i adet giriş için bir çıkış satırı gösterilmiştir. Örneğin ilk nesil sıralı PAL entegre devrelerinden PAL16R8 entegresinde 8 giriş, 8 çıkış, bir clock ve bir OE (Output Enable, çıkışı aktif eden kontrol girişi) terminali vardır. Ayrıca Lattice Semiconductor firması tarafından üretilmiş, Türetilmiş Dizi Lojik (GAL, Generic Array Logic) isminde PLD'ler de bulunmaktadır. PAL ve GAL marka isimleridir ve üreticisinin PLD ürün ailesini temsil etmektedir. PAL ve GAL ürün ailelerinde, birbirlerine muadil çok sayıda PLD entegre devresi bulunmaktadır [119, 120].

AHMET TURAN ÖZDEMİR - atozdemir@erciyes.edu.tr



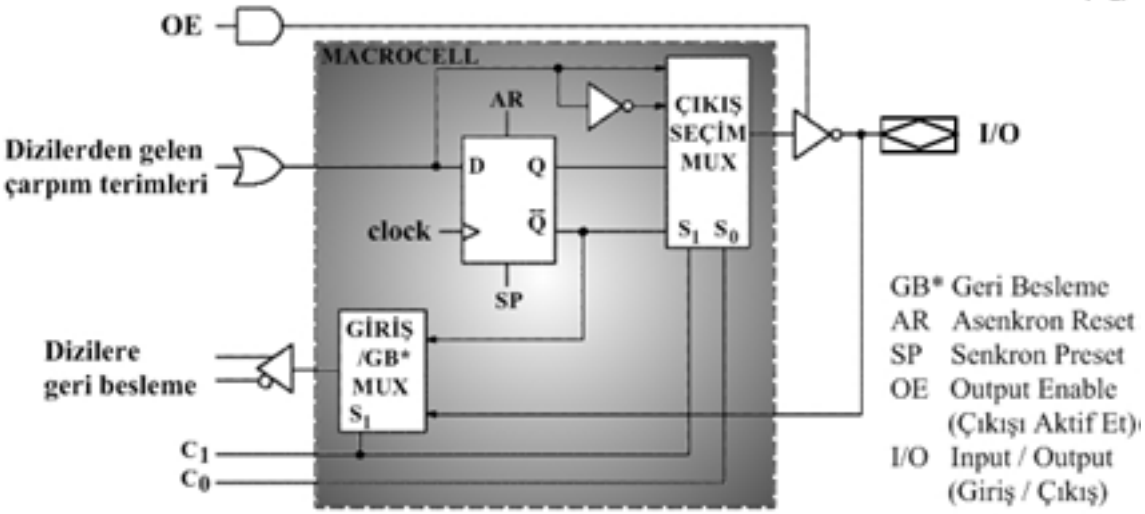
Şekil 6.1. Temel PAL mimarisi.

PAL'lerin kullanımı ile lojik devre tasarımlarında pek çok avantaj sağlanmıştır. Bu entegre devrelerin içerisinde daha fazla sayıda lojik eleman bulunması sebebi ile tasarımlar daha az sayıda entegre devre ile gerçekleştirilebilmektedir. Daha az sayıda entegre devre demek, daha küçük bir devre kartı, daha düşük bir maliyet ve daha az enerji tüketimi anlamlarına gelir. Ayrıca az sayıda entegre devre, daha az bağlantı anlamına gelir. Bu ise arıza arama ve test işlemlerini kolaylaştırır. PAL ile sağlanan en önemli avantajlardan biri de tasarımın güvenliğidir. Geleneksel 7400 serisi entegre devreler ile yapılan bir tasarım, bir mühendis tarafından kolayca analiz edilip kopyalanabilir. Tasarımın bir PAL yongası içerisinde oluşturulması ile kopyalanması önlenmektedir [119, 121, 122].

PAL aynı zamanda çok büyük bir tasarım esnekliği de sunmaktadır. Tasarımcı istediği sayısal tasarımı, devre elemanı bulup bulamayacağını düşünmeden, tek tip entegre devreler ile gerçekleştirebilmektedir. Bu tasarım esnekliği tasarım yapmayı daha karmaşık bir hale soksa da, üreticilerin sunmuş olduğu geliştirme yazılımları, tasarım sürecini daha da basitleştirmiş ve kısaltmıştır. Bu entegre devrelerin çok önemli diğer bir avantajı ise programlanabilir olmalarıdır. Bu sayede hataların giderilmesi veya tasarımın geliştirilmesi için devre kartının ve üzerindeki entegrelerin değiştirilmesine gerek kalmamaktadır [103, 119].

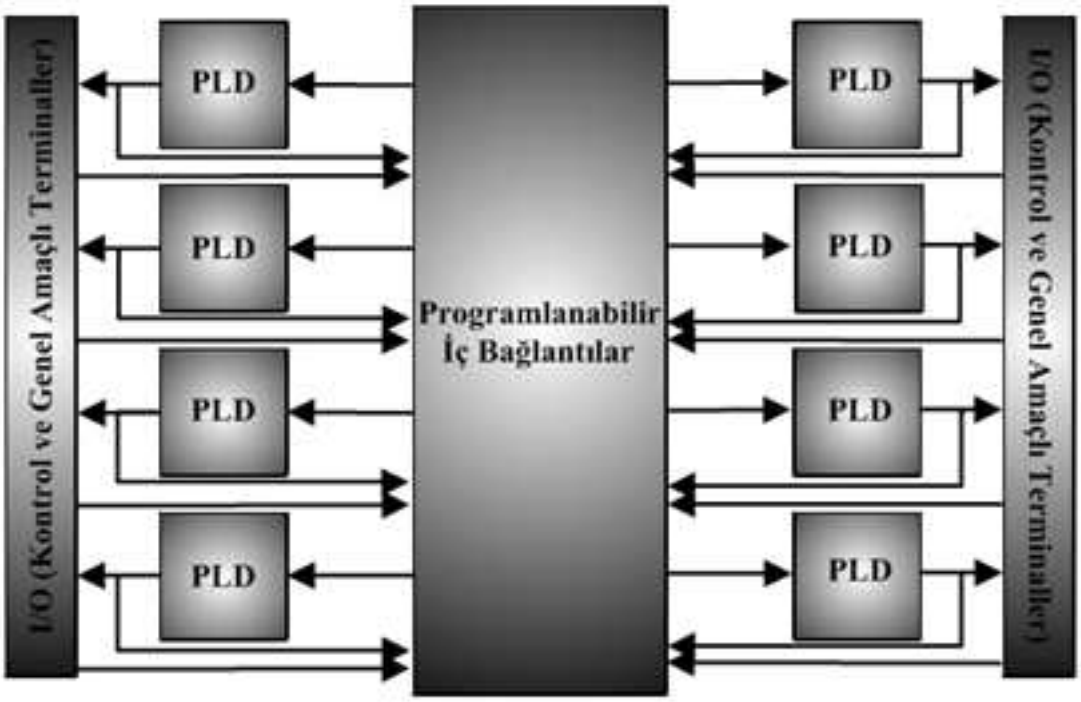
PLD entegrelerinin programlama yetenekleri, Macrocell'lerin de programlanabilir özellikte tasarlanması ile daha da geliştirilmiştir. Şekil 6.2'de GAL22V10 entegre devresi içerisindeki programlanabilir Macrocell'in içyapısı gösterilmektedir. Şekilden açıkça görüldüğü gibi seçici değişkenler (C_0, C_1) vasıtası ile hafıza elemanı bypass edilebilir; bu sayede aynı entegre devre kombinasyonel veya sıralı bir PLD olarak kullanılabilir. Yine çarpımların toplamı fonksiyonu

ya da hafıza değeri çıkışa aktarılabilir veya programlanabilir dizilere geri beslenebilir. Ayrıca çıkış terminali istenildiğinde bir giriş terminali olarak da kullanılabilir. Bu yüzden bu terminale I/O (Giriş/Çıkış) terminali denilmektedir. AR ve SP ile başlangıç veya şartlı değerlerin oluşturulması artık çok daha kolaydır. Bu yapı o kadar büyük bir tasarım esnekliği sunmaktadır ki, bazı özellikleri halen günümüz programlanabilir lojik mimarileri içerisinde bulunmaktadır. Mimarideki bu süreç, donanım geliştirme yazılımlarını da paralelinde geliştirmiştir [119, 120].



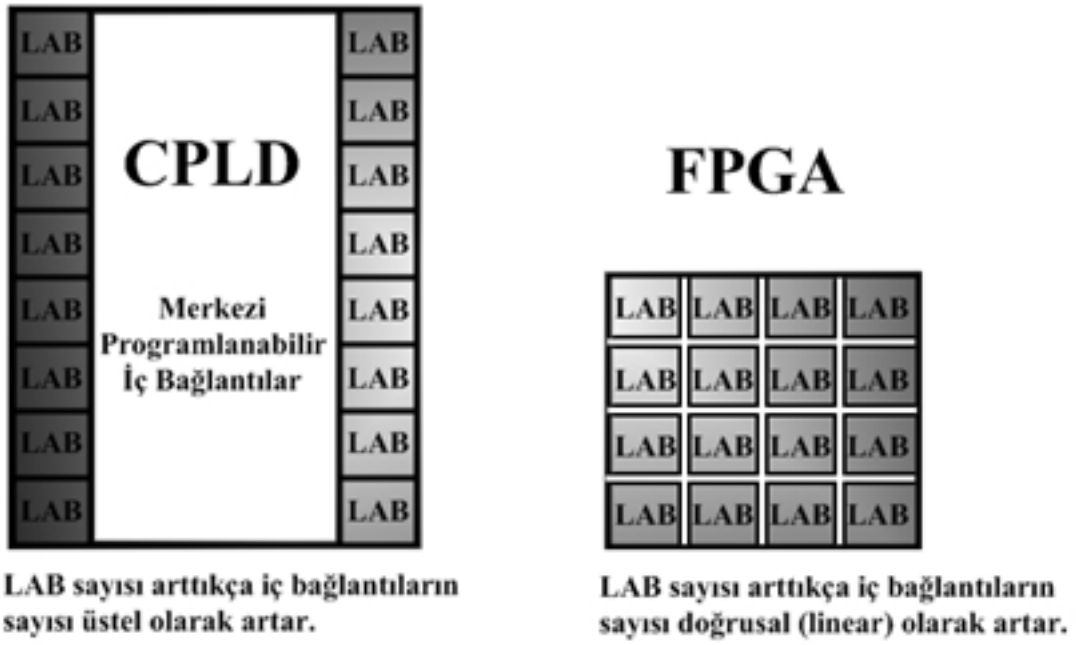
Şekil 6.2. GAL22V10 içerisindeki programlanabilir Macrocell.

Entegre devre teknolojisinde yaşanan gelişmeler ile yonga üreticileri daha büyük PLD entegreler yapmaya yönelmişler ve bunun sonucu olarak CPLD'ler markete çıkmıştır. CPLD'ler çok sayıda PLD'nin tek yonga içerisinde toplanmış hali olarak düşünülebilir. Şekil 6.3'te temel bir CPLD mimarisi verilmiştir. CPLD içerisindeki çok sayıdaki PLD'nin birbirleri ile olan bağlantıları da programlanabilir özelliktedir. CPLD içerisindeki PLD blokları, genellikle Lojik Dizi Bloğu (LAB, Logic Array Block) isimlendirilir. Her bir LAB içerisinde ortalama 4 ile 20 arası Macrocell bulunur [119].



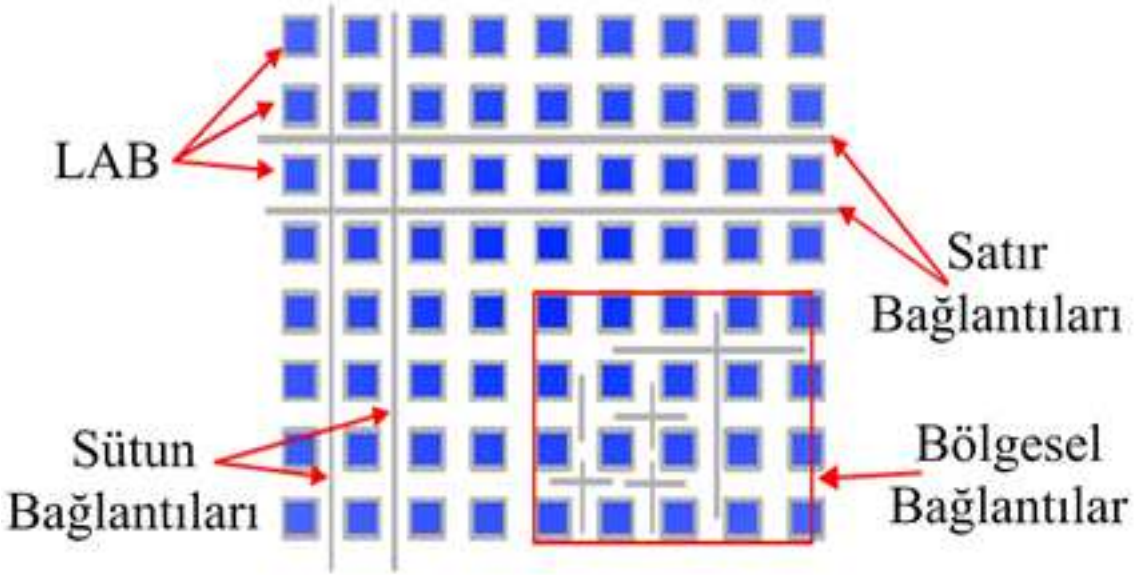
Şekil 6.3. Temel CPLD mimarisi.

CPLD'ler ile geleneksel PLD teknolojisi daha ileri bir noktaya taşınmıştır. Daha büyük ölçekli tasarımlar için daha yoğun CPLD yongaları üretilmiştir. Ancak bir tasarım ne kadar karmaşıklaştı ve iç bağlantıları ne kadar arttıysa, CPLD içerisindeki LAB'ları iyi organize edilmiş bir şekilde birleştirmek de o denli zorlaştı. Bu ise Merkezi Programlanabilir İç Bağlantılar bloğu üzerinden birbirleri ile bağlanabilecek maksimum LAB sayısında sınırlamalar meydana getirdi. Bu sınırlama, entegre devre üreticilerini merkezi bağlantı bloğu yerine satır ve sütunlar boyunca LAB'ların birbirlerine bağlanmasını mümkün kılacak yeni bir bağlantı tekniği oluşturmaya yöneltti. Şekil 6.4'te yüksek yoğunluklu bir CPLD içerisindeki LAB bloklarının, daha küçük bir alan içersinde nasıl etkin bir şekilde bağlanabileceği gösterilmiştir. Şekilden de görüldüğü gibi LAB blokları tıpkı bir şehir planında olduğu gibi, her birini paralel ve dik olarak geçen bağlantı yolları arasına yerleştirilmişlerdir. Bu yerleşim tekniği sayesinde hücrelerin birbirleri ile bağlantıları daha etkili bir hale getirilmiştir. CPLD'lerde artan LAB sayısı bağlantı yollarının sayısını üstel bir şekilde arttırırken, yeni bağlantı tekniği ile bağlantı yolları daha etkin kullanılmış ve bağlantıların artışı doğrusal bir düzeye çekilmiştir. Bu yeni bağlantı tekniği FPGA yongaları içerisinde etkin bir şekilde kullanılmaktadır [119, 120, 122].



Şekil 6.4. İç bağlantılar.

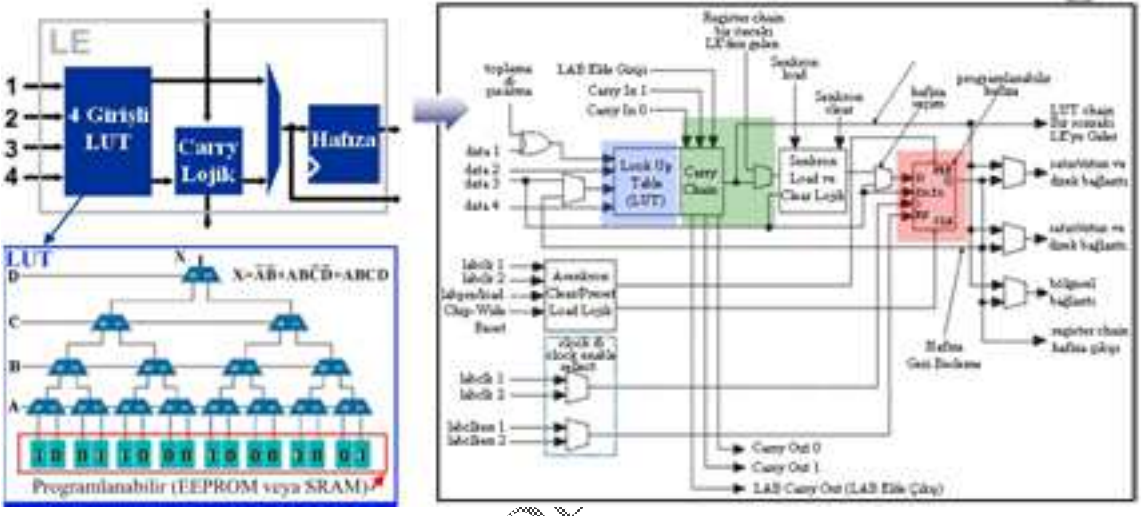
FPGA yongasının içinde yer alan LAB'lar Lojik Eleman'lardan (LE, Logic Element) oluşur ve tıpkı CPLD'de olduğu gibi bütün hücreler yeniden programlanabilir özelliktedir. CPLD ile FPGA arasındaki farklılıklardan en göze çarpanları, merkezi bağlantı yolları yerine satır-sütunlardan oluşan ve istenilen herhangi bir bölgede bağlantıyı mümkün kılan yollar ve farklılaşmış LAB bloklarıdır. FPGA yongası içerisindeki bağlantı tekniği Şekil 6.5'te verilmiştir [119, 120, 123, 124].



Şekil 6.5. FPGA iç bağlantıları.

CPLD'ler içerisindeki LAB'lar programlanabilir diziler ve Macrocell'lerden oluşurken, FPGA içerisindeki LAB'lar bir dizi yeni özellik ile daha iyi konfigüre edilebilen ve

kullanılamayan lojik kaynak sayısını azaltan, programlanabilir LE'lerden oluşur. Şekil 6.6'da bir LE, blok diyagram ve şematik gösterim olarak verilmiştir. LE'lerin bu avantajları FPGA yongalarını CPLD yongalarından daha kullanışlı hale getirmiştir. Her bir LE hücresi dört girişli bir giriş hafıza (LUT, Look Up Table), bir Elde Lojik (Carry Logic) ve bir çıkış hafıza bloğundan oluşur. LUT yöntemi ile fonksiyonlar kolayca üretilebilir ve saklanabilir bir hale gelmiştir. Ayrıca günümüz FPGA'ları gömülü çarpıcı, Faz Kilitli Döngü (PLL, Phase Locked Loop), gömülü hafıza, yüksek hızlı alıcı-verici blokları gibi gelişmiş birçok ilave donanım ile birlikte üretilmektedir [122, 123].



Şekil 6.6. LE içyapısı.

Üzerinde her türlü sayısal tasarımının oluşturulabileceği FPGA donanım geliştirme platformu, büyük ölçekli tasarımları hızlı ve ucuz bir şekilde ortaya koyabilme avantajı sağlar. Çünkü FPGA teknolojisi sayesinde her bir tasarım veya revizyon için yonganın tekrar üretilmesi zorunlu olmadığı gibi güçlü geliştirme yazılımları ile çok karmaşık tasarımlar dahi kısa sürelerde oluşturulabilmektedir. Her ne kadar Uygulamaya Özel Entegre Devrelerin (ASIC, Application Specific Integrated Circuit) yonga boyutu ve güç tüketimi FPGA'lardan çok daha küçük olsa da üretim süreçleri uzun ve maliyetlidir. FPGA yongalarında üretim süreç ve maliyeti, bilgisayar ortamında tasarlanan mimarinin sentezlenerek yonga üzerine yüklenmesi kadar kısa ve ucuzdur. Günümüzde içerisinde milyonlarca kapı elemanı olan FPGA yongaları üretilmektedir. Sonuç olarak FPGA yongaları, tasarım aşamasında veya az sayıda üretilecek olan ürünler için pratik bir uygulama geliştirme ortamı olmaktadır [125].

İşaret işleme uygulamalarında algoritmalar genellikle gelişmiş mikroişlemciler üzerinde gerçeklenir. Yüksek seviyeli özel amaçlı algoritmalar ise ya bir dizi Sayısal İşaret İşlemci (DSP, Digital Signal Processor) ya da ASIC yongalar üzerinde gerçeklenmektedir. FPGA

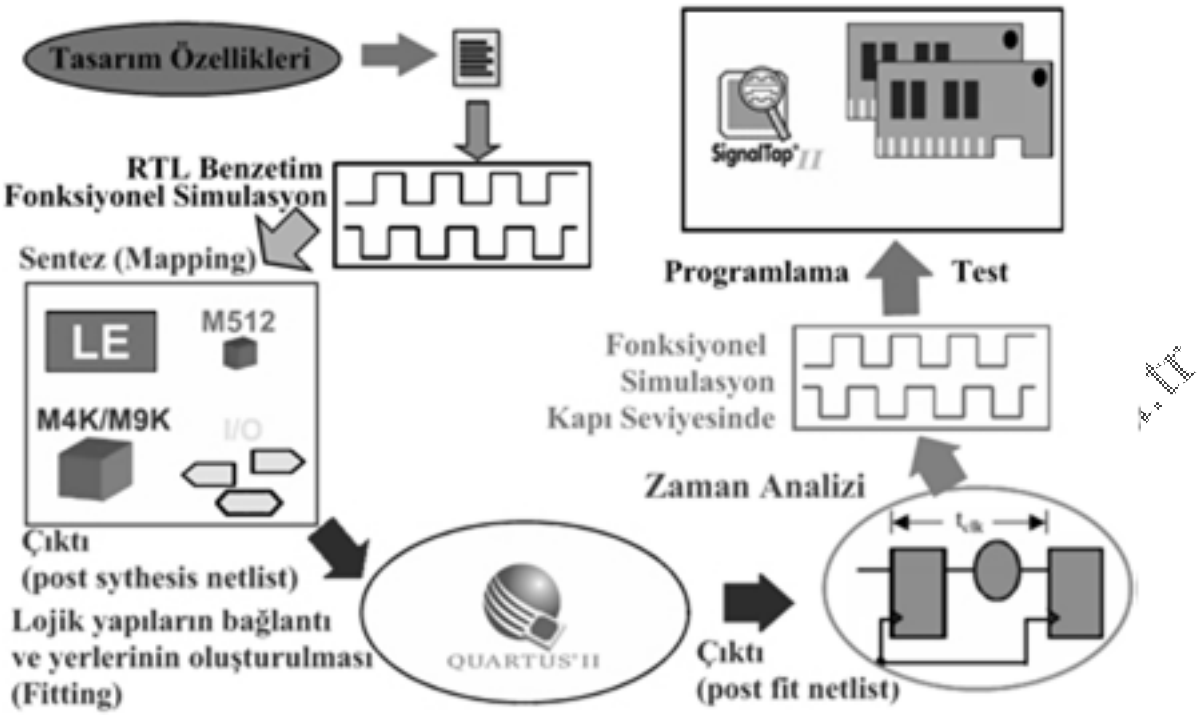
teknolojisindeki gelişmeler sinyal işleme alanında çalışan araştırmacılara yeni yaklaşımlar sunmaktadır. FPGA yongasının uygulamalardaki temel avantajı, tıpkı bir ASIC gibi özel bir amaç için tasarlanabilir olmasının yanında, ASIC'ten farklı olarak düşük geliştirme ve üretim maliyetlerinin yanında daha sonra yapılabilecek değişiklik ve eklentilere de imkân sağlamasıdır. Ayrıca FPGA optimal devre gereksinimlerini gerçekleme hususunda, tasarım esnekliğinin yanı sıra, gerek PCB alanı gerekse güç tüketimi bakımından geleneksel DSP yongalardan daha iyidir. ASIC'e ihtiyaç duyulan uygulamalarda eğer pazara çıkma süresi veya tasarım uyumluluğu önemli birer parametre ise FPGA burada iyi bir çözüm olabilmektedir. FPGA kapı dizilerinin esnekliği ve performansı hızlı tasarım süreçlerini mümkün kılmaktadır. Tasarım yenilenmek veya değiştirilmek istenildiğinde, tasarımın gerçekleştirildiği yonga veya PCB değiştirilmek yerine yeniden programlanır [123].

FPGA'lar içerisinde konfigürasyon bilgisi saklamadıkları için, ilk enerjilendiklerinde harici bir EEPROM, CPLD veya işlemci gibi bir dış birim üzerinden konfigürasyon bilgilerinin okunarak programlanmaları gerekmektedir. SRAM türü çok hızlı hafızalar ile programlama süresi minimuma indirilerek bu dezavantaj ortadan kaldırılmıştır. Bir FPGA aktif ve pasif olmak üzere iki şekilde programlanabilir. Aktif programlamada FPGA enerjilendiği zaman, bir dış hafıza içerisindeki programlama dosyası FPGA tarafından okunarak programlama işlemi otomatik olarak tamamlanır. Pasif programlamada ise programlama süreci, işlemci gibi zeki bir çevresel donanım tarafından yapılır. İşlemci EEPROM'da veya başka bir konfigürasyon biriminde saklanan programın FPGA'ya nasıl ve ne zaman yükleneceğini kontrol eder. Ayrıca tasarım esnasında bilgisayarda oluşturulan tasarımların FPGA'ya yüklenmesi için JTAG arayüzü üzerinden bir kablo ile de programlanması mümkündür. Doğruluğu onaylanmış bir tasarım ASIC'e dönüştürülebilir, üretici firmalar adetli üretimler için konfigürasyonu sabitlenmiş FPGA'lar da üretmektedirler. Bu şekilde ASIC haline getirilmiş son ürünlerde hem birim adet başına maliyet düşer, hem de harici bir dış hafıza ve programlama donanımlarına ihtiyaç kalmaz. Bu da programlama için ihtiyaç duyulan süre sıkıntısını da ortadan kaldırır [119, 120, 123, 124].

6.1. FPGA ve Donanım Geliştirme Yazılımları

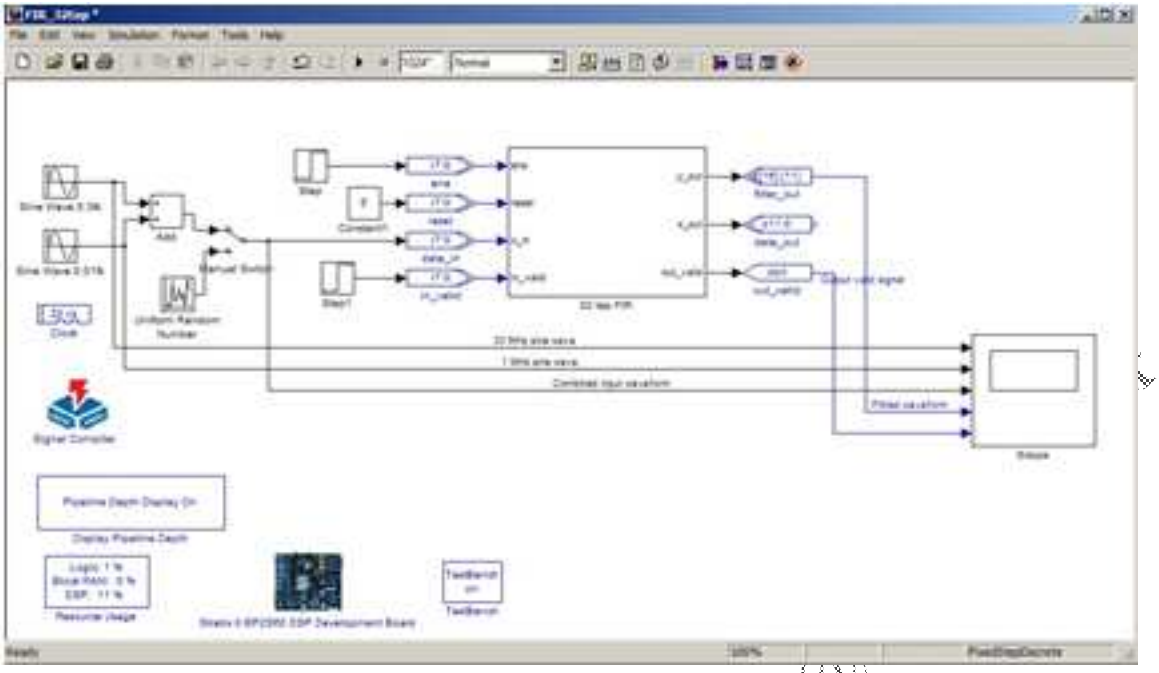
Bu tez çalışmasında Altera firmasının Cyclone III FPGA yongası kullanmıştır. Altera firmasının kendi ürettiği FPGA'ları için tasarımdan, programlama ve teste kadar bütün süreçlerinde kullanılabilen Quartus II isminde bir yazılımı bulunmaktadır. FPGA ile tasarım süreci, tasarım özelliklerini belirlemekle başlar. Tasarımın özelliklerine göre, davranışsal veya

lojik yapılar, gerek bazı grafik ara yüzleri gerekse Donanım Tanımlama Dili (HDL, Hardware Description Language) kullanılarak Register Transfer Seviyesinde (RTL, Register Transfer Level) oluşturulur. Genellikle bu işlemin ardından Mentor Graphics veya ModelSim gibi yazılımlar kullanılarak oluşturulan yapının RTL fonksiyonel benzetimi yapılır. Fonksiyonel benzetim sadece lojik yapının doğruluğunu test etmektedir, FPGA içerisindeki zaman gecikmeleri bu benzetimde dikkate alınmaz. Bir sonraki adımda tasarım seçilen FPGA yongasının sahip olduğu donanım kaynak ve özelliklerine göre sentezlenir. Tasarım, sentez işlemi süresince arzu edilen alan, zaman ve performans kriterlerini karşılayacak şekilde optimize edilir. Sentez editörü olarak FPGA üreticilerinin kendi yazılımları kullanılabildiği gibi Mentor Graphics Precision Synthesis, Synplicity Synplify, Syplify Pro veya Synopsis Design Compiler gibi diğer sentez editörleri de kullanılabilir. Sentez işlemi tamamlandıktan sonra "post synthesis netlist" olarak isimlendirilen bir dosya üretilir. Sentez işlemi için hangi yazılım kullanılırsa kullanılsın, üretilen bu dosyanın FPGA yongası içerisinde hangi bağlantılar ile oluşturulacağını belirlemek için FPGA üreticisinin kendi programının kullanılması gerekmektedir. Quartus II Fitter yazılımı, post synthesis netlist dosyası içerisindeki lojik ifadeleri seçerek hedef donanım içerisinde lojik yapılara dönüştürür ve FPGA içerisindeki mevcut bağlantı yollarını kullanarak bu yapıları birbirlerine bağlar. Quartus II Fitter yazılımı, tasarımcının belirlediği veya varsayılan alan, performans, zaman ve güç kıstaslarına göre "post fit netlist" isminde bir dosya üretir. Bu dosya daha sonra bir dizi işlemden daha geçirilir. İlk olarak Quartus II TimeQuest isimli yazılım ile oluşturulan donanımın modelinin zaman kriterlerini karşılayıp karşılamadığı test edilir. Daha sonra, arzu edilirse bu donanım modeline kapı seviyesinde benzetim uygulanır. Bu benzetim RTL benzetimden farklı olarak bağlantı yollarından kaynaklanan gecikmeleri de dikkate alır. Dolayısıyla FPGA içerisinde oluşturulacak olan donanımın, FPGA yongasına yüklendiği zaman doğru bir şekilde çalışıp çalışmadığı bu sayede test edilebilir. Artık "post fit netlist" dosyasından FPGA'nın programlanmasına yarayan konfigürasyon dosyası üretilebilir. Bu dosya JTAG ara yüzü ile SRAM hafızaya yüklenir. Bu işlemin ardından arzu edilirse Quartus II SignalTap programı ile FPGA'nın içerisine gömülmüş olan donanımın fiziksel ortamda doğru çalışıp çalışmadığı da kontrol edilebilir. Şekil 6.7'de bütün bu süreç özetlemiştir [119, 126].



Şekil 6.7. FPGA ile tasarım süreci.

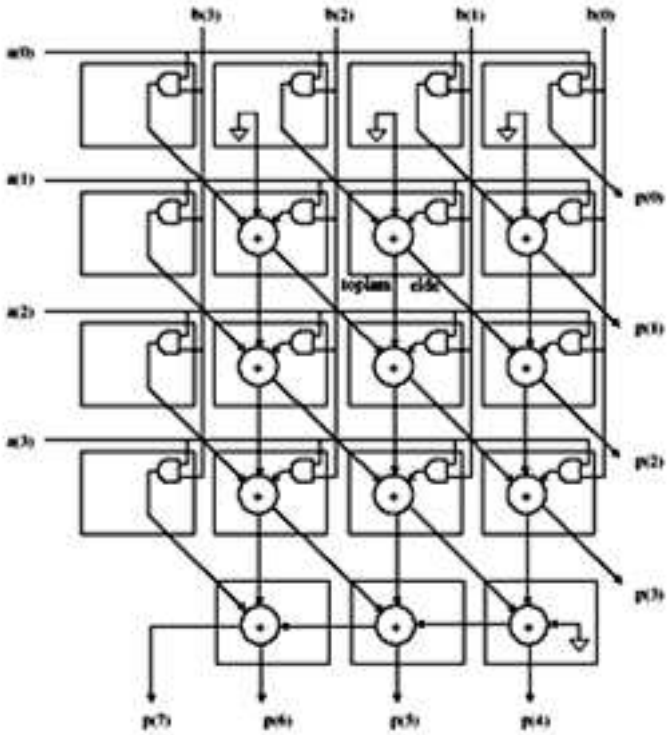
Sinyal işleme denildiğinde gerek akademi çevreleri gerekse ileri teknoloji kuruluşlarının sıklıkla başvurduğu bir araç olan Matlab, DSP ve FPGA tasarımcıları için de geliştirme araçları sunmaktadır. Matlab Simulink blok tabanlı tasarımda oldukça kullanışlı bir geliştirme platformudur. Simulink gerek Texas İnstruments DSP yongaları için gerekse Altera ve Xilinx FPGA'ları için DSP tasarım geliştirme ortamı sunmaktadır. Şekil 6.8'de, Aletra DSP Builder ile Simulink ortamında modellenmiş bir filtre tasarımı görülmektedir. Altera DSP Builder programı bu model dosyasından HDL kod dizini oluşturarak, tasarımın FPGA üzerinde donanım olarak gerçekleştirilmesine olanak sağlar. Altera DSP Builder ile Simulink'in zengin kütüphanesi içerisindeki algoritma ve modeller, FPGA ile tasarım yapan uygulamacıların kullanımına sunulmuştur. İşlemsel doğruluk ve optimum algoritmayı belirlemede Matlab ve Simulink kütüphanelerinin sunduğu çözümlerin HDL kodlarına dönüştürülerek donanımlara aktarılması çok önemli bir avantajdır [127-129].



Şekil 6.8. Altera DSP Builder ile Simulink ortamında tasarlanmış bir model.

FPGA, tasarımcılara işlemcilerin sahip olmadığı bir dizi avantaj sağlar. Bunların belki de en önemlilerinden biri, paralel çalışan çok hızlı donanımların FPGA içerisinde oluşturulabilmesidir. Çünkü FPGA mimarisi, sıralı işlem birimlerinden farklı olarak paralel işlem yeteneğine sahiptir. Çok adımda yapılacak bazı işlemler, FPGA içerisinde oluşturulacak uygun bir lojik devre ile tek kenar tetiklemesinde yapılabilmektedir. Şekil 6.9'da çarpma işlemini sadece kapıların yayılım gecikmesi kadar bir sürede tamamlayabilen bir paralel çarpıcı devresi görülmektedir. Burada a ve b girişlerine 4 bitlik çarpım terimleri paralel olarak uygulanır uygulanmaz cevap üretilir. Bu tip çalışmaya "paralel çalışma" adı verilir ve işlem süresi kapı elemanlarındaki toplam yayılım gecikmesi kadardır. Bu yapı, tasarımcı tarafından HDL kodları veya hiç kod yazmadan sadece şematik editör ile tasarlanabilir. Çok sayıda HDL programlama dili vardır, bunlardan en çok kullanılanları Çok Hızlı Entegre Devre Donanım Tanımlama Dili (VHDL, Very High Speed Integrated Circuit HDL) ve Verilog'dur [120, 124, 130].

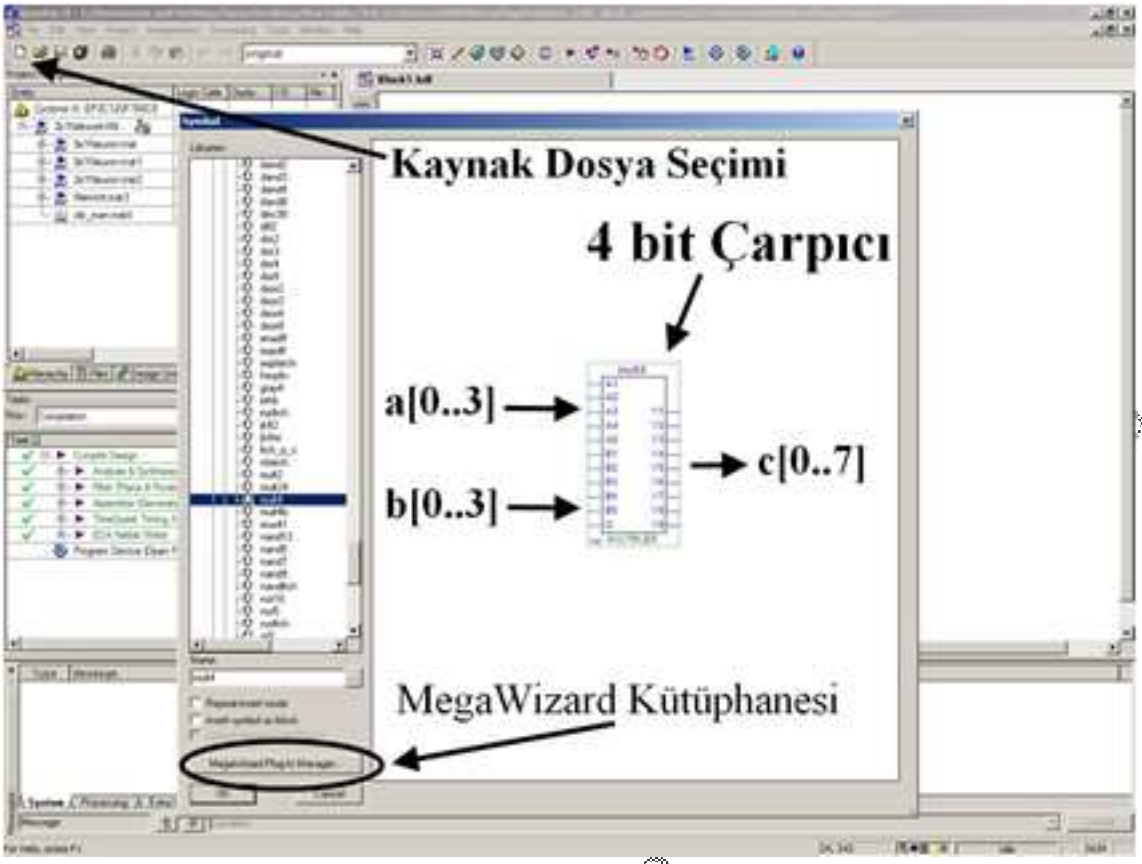
AHMET TURAN ÖZDEMİR (Anadolu Üniversitesi)



Şekil 6.9. Dört bit paralel çarpıcı devresi.

FPGA tasarım editörleri tasarımcıya farklı esneklikler sağlamaktadır. Örneğin Şekil 6.9'da gösterilen 4 bitlik paralel çarpıcı devresi, Altera Quartus II FPGA editörünün kütüphanesi içerisinde önceden tanımlanmış bir elemandır. Bu yapı aşağıdaki gibi bir VHDL kod dizini oluşturulabileceği gibi, Şekil 6.10'da verilen şematik editörden sembol şeklinde de kullanılabilir.

```
1  ------------------------------------
2  library ieee;
3  use ieee.std_logic_1164.all;
4  use ieee.std_logic_arith.all;
5  ------------------------------------
6  entity multiplier4 is
7  port ( a, b: in signed (3 downto 0);
8  prod: out signed (7 downto 0));
9  end multiplier4;
10 ------------------------------------
11 architecture behavior of multiplier4 is
12 begin
13 prod <= a * b;
14 end behavior;
15 ------------------------------------
```



Şekil 6.10. Dört bit paralel çarpıcının şematik sembolü.

Gerek Simulink gerekse FPGA tasarım editörlerinin sunmuş olduğu güvenilir modellere ulaşmak ve bunları tasarımlarda kullanmak çok kolaydır. Şekil 6.10'da daha gelişmiş modellerin bulunduğu MegaWizard kütüphanesine erişim ikonu gösterilmiştir. MegaWizard kütüphanesi ile fourier dönüştürücü veya filtre gibi daha kompleks algoritmaları yerine getiren modellere ulaşmak ve bunları tasarımlar içerisinde kullanmak mümkündür. Ayrıca IP Core isimli başka bir kütüphanede sayısal işaret işlemeye ait daha birçok farklı algoritmanın da model dosyalarına ulaşmak mümkündür. Bazıları ücret karşılığında satın alınan bu modelleri tasarımcı kendi projelerinde kullanabilir. Ayrıca Altera Quartus II editörü SoPC Builder isimli Altera'nın başka bir yazılımı ile birlikte çalışmakta ve FPGA içerisinde kullanıcıya özel işlemciler tasarlamasına müsaade etmektedir. İstenilen sayıda dış birim (giriş çıkış portları, VGA sürücü, i2c, serial, eternet, usb vs.) ile uygulamaya özel işlemcilerin tasarlanması ve bu tasarımların FPGA içerisinde kullanılması mümkündür [131].

6.1.1. FPGA ile Tasarımın Avantajları

Bugün tasarım şirketlerinde çalışan birçok mühendis, kısa zaman içerisinde sınırlı kaynaklar ile yeni şeyler tasarlarken, işverenler onlardan yüksek performans bekler ve ürettikleri şeyin farklılıklarla pazarda söz sahibi olmasını isterler. Ayrıca çok hızlı değişen teknoloji müşterilerin isteklerinin de farklılaşmasına sebep olmaktadır. Bir mühendis tasarımında yeni bir teknolojiyi kullanmayı düşündüğünde, kullanacağı teknolojinin mevcut en iyi olup olmadığını ve bu yatırımın iyi bir geri dönüşü olup olmayacağını düşünmek zorundadır. Bu zorunluluk günümüz tasarım mühendislerinde ciddi anlamda strese neden olmaktadır. 20 yıldan uzun bir süredir, FPGA gibi programlanabilir lojik devreler bu sıkıntıların giderilmesi noktasında pek çok katkıda bulunmuştur. FPGA devreler teknolojiye liderlik ederek esnek, uyarlanabilir ve düşük maliyetli çözümler ile ürünün pazara çıkma süresini kısaltmaktadır. Sahada çalışan bir FPGA yongasına dahi uzaktan erişilerek donanım güncellemesi yapılabilir. Bu işlem tasarımcının üzerindeki son ürün stresini çok önemli ölçüde azaltan bir avantajdır. Ayrıca Quartus II gibi FPGA tasarım geliştirme platformları takım çalışması için uyumlu yazılımlardır. Mühendisler projenin değişik birimleri üzerinde çalışabilir ve parçalar ortak platformda birleştirilerek proje tamamlanabilir.

6.1.2. Kullanım Alanları

FPGA ile tasarımda uygun donanımların kütüphanelerden hazır olarak kullanılması mümkündür. Aynı sayıda çarpım terimini içeren bir çarpma işlemi, mikroişlemci üzerinde yapıldığında çarpım sonucunun elde edilmesi için daha fazla sayıda saat darbesine ihtiyaç duyacaktır. Bazı DSP ve ASIC yongalarda bu tür işlemlerin minimum sürelerde yerine getirilmesi için özelleşmiş çarpıcı birimler kullanılmaktadır. Ancak işlemcilerde çarpım işlemi sayısı birden fazla olduğu durumlarda, bu özelleşmiş hızlı çarpıcının paylaşılması yani ikinci çapma işleminin yapılabilmesi için birinci çarpma işleminin tamamlanması beklenecektir. Dolayısıyla paralel işlemin sunduğu avantajlar DSP yongalarla sağlanamamaktadır. Bununla birlikte FPGA, uyarlanabilir donanım yapısı, işlemsel kabiliyeti, tasarım süreçlerindeki hızı, düşük üretim maliyetleri ve FPGA tasarım editörünün sunduğu kolaylıklarla tasarımcıların ilgi odağı olmaktadır. FPGA sadece temel aritmetik işlemleri uygulayan mimarilerde değil sayısal tasarımın ihtiyaç duyulduğu pek çok uygulamada da kullanılmış tümleşik bir tasarım geliştirme platformudur [124, 130, 132-135].

ASIC yongaların tasarım ve üretim süreçleri çok uzun ve maliyetleri çok yüksek olduğu için bir ürünün ortaya konulması kolay değildir. Bu sebeple ASIC yongalarda ilk örnek tasarımlar sıklıkla FPGA'lar üzerinde gerçekleştirilmektedir. Bütün sayısal tasarımları FPGA üzerinde

AHMET DURANDEMİR, ahmetdurandemir@yahoo.com.tr

gerçekleştirmek mümkündür. Analog yapılar ayrık bir ön veya sonlandırıcı devre ile modellenebilir. Analog ön devreli sayısallaştırılmış sistemlerde yüksek hızda doğru verilerin elde edilmesi için birçok yapı kullanılabilmektedir. FPGA yüksek hız, esnek donanım, programlanabilir yapısı ve düşük güç tüketimi ile devre gerçeklemelerinde önemli bir avantaj sunmaktadır [128].

ASIC yongalar üretildikten sonra üzerlerinde bir değişiklik yapmak mümkün değildir. İşlemci tabanlı sistemlerde ise algoritmaları değiştirmek mümkün iken, iç donanımlar sabittir. Ayrıca, PCB üzerine dizilmiş olan bir sistemde donanımsal değişiklikler yapmak hem zaman alıcı hem de külfetlidir. Yapılacak her türlü donanımsal değişiklik yeni bir PCB ile gerçekleştirilir.

İşlemci tabanlı tasarımlar yapan mühendisler bir sistemi tasarlarken, ihtiyaçları ve bu ihtiyaçları karşılayacak çözümleri dikkatle tanımlayarak, arzu edilen özelliklerde bir mikroişlemci belirler. Fakat genelde arzu edilen özellikleri birebir karşılayan bir model bulunamaz. Mevcut yapılar ya ihtiyacın altında ya da ihtiyacın üstündedir. Fakat FPGA teknolojisi ile istenilen özellikte soft-core mikroişlemci tasarlamak mümkündür. FPGA üzerinde analog yapılar oluşturulamamakla birlikte ön ve sonlandırıcı analog/sayısal dönüştürücüler ile karma bir sistem işler hale getirilebilmektedir. Fakat bu, analog yapıların programlanabilir elemanlar ile oluşturulamayacağı anlamına gelmemektedir. Alan Programlanabilir Analog Diziler (FPAA, Field Programmable Analog Arrays) bazı analog yapıların yonga üzerinde gerçeklenmesine imkân tanımaktadır. FPGA ve FPAA yongaları birlikte kullanıldığı uygulamalar da bulunmaktadır [136, 137].

6.2. YSA'nın Sayısal Sistem Modellemesi

Biyolojik beyin benzetimi olan YSA matematiksel bir modele sahiptir. Bilgisayarlar üzerinde benzetimleri yapılabildiği gibi devre gerçeklemeleri de mevcuttur. YSA'nın doğasındaki paralellik, gerçek anlamda sadece donanımsal olarak oluşturulabilmektedir. YSA benzetim uygulamaları için Neumann tipi bilgisayar mimarileri sıklıkla tercih edilmektedir. Fakat bu tip benzetimlerde sinir ağının boyutu arttıkça benzetim zamanı da artmaktadır. Dahası sıralı işlem mantığı YSA'nın doğasına aykırıdır.

YSA'ların etkinliklerini arttırmak ve sahada kullanımını mümkün kılmak için bilgisayardan bağımsız donanımsal modellerinin oluşturulması üzerine çalışmalar yapılmaktadır. Bu çalışmalar nöro donanımlar (neuro hardware) başlığı altında incelenebilir. Nöro donanımlar analog, sayısal ve karma yöntemlerle, VLSI veya ASIC yongalar üzerinde oluşturulmuştur [37]. Analog YSA gerçeklemeleri düşük maliyet ve yüksek hız gibi avantajlarına rağmen

zayıf hata bağışıklıkları ve sabit mimarileri büyük birer dezavantajdır. Diğer taraftan sayısal YSA gerçeklemeleri ise çok fazla donanım kaynağı kullanan büyük ölçekli çarpıcılar ve NAF bloklarına ihtiyaç duyarlar. Bu sebeplerden dolayı genellikle YSA'lar, tek bir çarpıcı ve NAF bloğunun paylaşılarak kullanıldığı güçlü mikroişlemciler veya DSP'ler üzerinde gerçekleştirilirler. Fakat mikroişlemciler ve DSP'lerin sıralı işlem özelliği, YSA'nın paralel mimarisinin gerçekleştirilmesi için uygun bir geliştirme ortamı sunamaz. VLSI ve ASIC seçenekleri ise gerek çok uzun tasarım süreleri ve çok pahalı ilk üretim maliyetleri, gerekse sundukları sınırlı esneklikleri gibi dezavantajları yüzünden YSA donanım gerçeklemeleri için çok uygun değillerdir. FPGA gerçeklemeleri ise tasarımcıya ASIC ve VLSI gerçeklemelerine göre daha düşük bir maliyet, hızlı ilk örnekleme, uyarlanabilir donanım, güçlü yazılım geliştirme araçları desteği ve dağıtılmış paralel mimari gibi avantajlar sunar [38].

FPGA tabanlı YSA gerçeklemelerinde en önemli kıstas, mimarinin nasıl etkin bir şekilde kullanılacağıdır. İşlemci elemanlarda öncelikle girişler ilgili ağırlıklarla çarpılır ve daha sonra da bu çarpımlar toplanarak aktivasyon fonksiyonuna uygulanacak sinaptik çıkış üretilir. Çarpma işlemi yüzünden ihtiyaç duyulan alanın artması, FPGA tabanlı YSA'nın boyutunu belirleyen en önemli etkenlerden birisidir. Stephen Bade tarafından geliştirilen yetenekli bir look-up-table (LUT) yöntemi ile önemli bazı başarımlar sağlanmıştır [138]. . Bu yöntem yüksek seviyede yoğunlaştırılmış bir işlemci eleman oluşturmak için, doğrusal olmayan aktivasyon fonksiyonunun davranış karakteristiğini bir hafıza elemanı ile gerçekleştirmektedir. Ayrıca H. Hikawa [139] işlemci elemanlar için aktivasyon fonksiyonu olarak üç değerli bir fonksiyon önermiştir. Fakat bu mimariler bit hassasiyetindedirler ve yüksek hassasiyet sağlayan 32 bit kayan noktalı nümerik tanımlamalar ile kullanılamazlar ve düşük bitlerle ifade edilen düşük doğruluklu sayı formatları ile kullanılabilirler. Bu sınırlama ile çarpma işlemleri bit dizileri çarpımı şeklinde standart ikili sayı aritmetiği ile yapılmaktadır. Analog ve sayısal gerçeklemelerdeki karmaşıklıkları gidermek amacıyla araştırmacılar darbe modlu (pulse-mode) mimari kullanarak YSA donanımı geliştirme üzerine çalışmaktadırlar. Darbe modlu mimarinin avantajları, darbe modülasyon teknikleri ile birleştirilmiş temel sayısal devreleri kullanan sinaptik çarpıcı yapının, frekans modülasyon tekniklerini kullanması durumunda Direkt Dijital Frekans Sentezleyici (DDFS) ile çarpım sonucu üretilebilmesi ve doğrusal olmayan aktivasyon fonksiyonlarının en basit gerçeklemesi olmasıdır. Dezavantajı ise sinaptik ağırlıkların -1.0 ile +1.0 arasında sınırlı bir aralıkta değişmesidir. Küçük farklar içeren sinyalleri sınıflandırmak için eğitilmiş çok katmanlı bir sinir ağının, bu küçük faklılıkları kuvvetlendirmek için büyük ağırlıklara sahip olması

gerekmektedir. Bu yüzden küçük ağırlık aralığı büyük bir problemdir. Hikawa bu yapıyı geliştirerek parçalı doğrusal aktivasyon fonksiyonlu bir darbe modlu işlemci eleman önermiştir [140]. Önerilen yapıda aktivasyon fonksiyonu giriş sayısından etkilenmemektedir. Ağırlık değerlerinde de herhangi bir sınırlandırma söz konusu değildir. Fakat bu yapı sadece sabit noktalı tanımlamayla binary formatta oluşturulabilmektedir.

Literatürde yapılan çalışmaların yanında piyasada da uygulama türüne veya kullanılan nümerik tanımlama tipine göre değişik seçeneklerde üretilmiş ticari nöro donanımlar bulunmaktadır. Nöro donanımların etkili bir şekilde kullanılabilmesi için tasarım tipine uygun bir sınıflandırma probleminde kullanılması gerekir. Nöro donanımlar sistem mimarisi, paralellik derecesi, çoklu işlemci kullanılıyor ise her bir işlemcinin sorumlu olduğu ağ bileşeni, haberleşme şekli ve nümerik tanımlama tipine göre sınıflandırılabilir. Nöro donanımlar bağımsız işlem birimleri halinde çalışabildikleri gibi bir bilgisayar ile birlikte hızlandırıcı donanım olarak kullanılan şekilleri de mevcuttur [141]. Marketteki bazı ticari nöro donanımlar Tablo 6.1'de verilmiştir.

Her ne kadar Neumann tipi işlemcilerde çok önemli performans iyileştirmeleri sağlansa da sıralı işlem, gerçek zamanlı cevabı ve ağın eğitimini uzatmaktadır. Nöro donanımlar sahip oldukları paralel işlemci elemanlar vasıtası ile cevap süresi ve ağın eğitimini kısaltmaktadır. Çok hızlı işlem gücünün gerekli olmadığı uygulamalarda YSA modelleri genellikle bilgisayar üzerinde yazılımsal olarak çözümlenmektedir. Nöro donanımlar ASIC olarak sayısal veya analog yapılarda, sabit veya kayan noktalı nümerik tanımlamalar kullanılarak yapılmıştır. Nümerik hassasiyetin önemli olduğu uygulamalarda kayan noktalı nümerik tanımlamalar kullanılırken, hızın önemli olduğu uygulamalarda ise sabit noktalı nümerik tanımlamalar kullanılır. Sabit noktalı nümerik sayı formatı hızı arttırdığı gibi maliyetleri önemli ölçüde azaltmaktadır. Bununla birlikte nümerik hassasiyette de önemli kayıplara sebep olmaktadır.

Tablo 6.1. Marketteki nöro donanımlar [126].

Nöro-donanım	Tipi	Algoritma	Performans	İşlemci Eleman
CNAPS	Nöro-Chip	Geri Yayılım SOM	1.6 GCPS	64
SYNAPSE-1	Nöro-Chip	Geri Yayılım Hopfield Ağı	33 MCPS	
Hitachi WSI	Nöro-Chip	Geri Yayılım	150 MCPS	144
Ni 1000	Hızlandırıcı Donanım	Geri Yayılım RBF Ağı	8.2 GCPS	
NT 6000	Hızlandırıcı Donanım	Geri Yayılım Kohonen Ağı	2 MCPS	
IBM ZISC	Hızlandırıcı	RBF Ağı		36

AHMET TURAN ÖZDEMİR aturan@erciyes.edu.tr

Nöro donanımlarda bir performans ölçüsü olarak Saniyede Hesaplanan Bağlantı (CPS, Connections Per Second) değerleri kullanılır. CNAPS (Connected Network of Adaptive Processor) The Adaptive Solution firması tarafından üretilmiş bir nöro bilgisayardır. Kaskat olarak bir veri yolu üzerinden birbirlerine bağlanmış 64 adet işlemsel elemandan oluşan N6400 nöro-chip yongası, içlerinde geri yayılım veya SOM algoritmalarının da bulunduğu birçok modelin gerçekleştirilebildiği bir donanımdır. CNAPS içerisinde yapay sinir ağının ağırlık değerlerinin saklandığı bir de hafıza elemanı bulunmaktadır. CNAPS ilave N6400 yongaları ile ağın büyütülmesine de imkân tanımaktadır. Geri yayılım algoritması ile 8 veya 16 bit ağırlık değerleri için 1.6 GCPS hızına sahiptir [126, 141, 142].

SYNAPSE 1 (Sythesis of Neural Algorithms on Parallel Systolic Engine) Siemens'in MA-16 nöro-chip yongası ile çalışan bir nöro bilgisayardır. Motorola MC68040 işlemcisi tarafından kontrol edilen 8 adet MA-16 yongasından oluşan bu donanım, aktivasyon fonksiyonu cevabını LUT yöntemi ile gerçekleştirmektedir. İçlerinde geri yayılım ve Hopfield algoritmalarının da bulunduğu birçok algoritmayı 16 bit sabit noktalı nümerik tanımlama ile gerçekleştirir [126, 141, 143].

Hitachi WSI 144 adet işlemci elemandan oluşan, 16 bit uzunluklu ağırlık değerlerine ve 9 bit uzunluklu aktivasyon fonksiyonu tanımlamalarına sahip olan bir nöro bilgisayardır [126, 144].

Nöro bilgisayarların yanında, daha ucuz maliyetli ve gelişmiş yazılım desteğine sahip olan ve bilgisayarların ana kartları üzerindeki ISA veya PCI genişleme yuvalarına monte edilen hızlandırıcı nöro donanımlar da bulunmaktadır. Bu donanımlardan birkaçı aşağıda verilmiştir.

Ni 1000, Nester/Intel tarafından üretilmiş ve örüntü sınıflandırmada kullanılan çok hızlı bir hızlandırıcı donanımıdır. 16 bit bir mikroişlemci ve bilgisayar iletişim hattı bulunan bu hızlandırıcı donanım üzerinde, geri yayılım ve RBF algoritmaları koşturulabilmektedir. Her biri 5 bit uzunluklu 256 adet özellik vektörünü, 33 sınıfa ayırabilen bu donanım saniyede 33.000 örüntüyü sınıflandırabilmektedir [126, 145].

NT 6000 geri yayılım ve Kohonen algoritmalarının koşturulabildiği bir hızlandırıcı donanımdır. Bu donanım, eğitim algoritmasının koşturulması ve giriş verilerinin uygun bir

AHMET TURAN ÖZDEMİR, aturan@erciyes.edu.tr

şekilde donanıma beslenmesi için genellikle bir DSP işlemci ile birlikte kullanılmaktadır. Hızı tipik olarak 2 MCPS değerindedir [126, 144].

ZISC (Zero Instruction Set Computer) IBM firması tarafından üretilmiş bir nöro-chip yongasıdır ve özel bir eğitim algoritması ile eğitilen RBF tipi ağların, 36 işlemci eleman ile oluşturulmasına imkân tanımaktadır. 16 ZISC yongasından oluşan ve toplam 576 işlemci elemana sahip, ISA uyumlu bir hızlandırıcı kart modeli de mevcuttur. 64 adet işlemci eleman bulunan bir hızlandırıcı, 8 bit uzunluklu giriş verisi için saniyede 165.000 örüntüyü sınıflandırabilir [126, 144].

Geçmişte araştırmacılar YSA'nın büyüklüğünün performansa etki eden en büyük parametre olduğunu düşünürlerdi. 1980'lerde nöro bilgisayar tasarımlarında temel mantık çok sayıda genel amaçlı işlemcinin paralel bağlanarak, büyük bir ağın hesaplama yükünün paylaştırılmasıyla hızlı bir şekilde işlemlerin yapılmasının sağlanmasıydı. 1990'larda CAM-Brain projesi ile YSA zekâsının sadece ağın boyutu ile tanımlanamayacağı ortaya konulmuştur [146]. Bugün YSA araştırmacıları tarafından ortaya konulan genel fikir, YSA başarımında ağ mimarisi ve iç bağlantıların önemli birer parametre olduğudur.
Ticari nöro donanımlar ASIC mimariye sahip olduklarından ve genel anlamda gerçek zamanlı uygulamalara çözüm olarak geliştirildiklerinden, çok özel uygulamaların dışında yaygın bir kullanıma sahip olamadılar. Ayrıca gerek maliyet, gerek temin edilebilirlik açılarından da pratik uygulamalarda bir avantaj oluşturamadılar ve çoğunlukla bilimsel araştırma projelerinde sınırlı bir uygulama alanı bulabildiler. FPGA teknolojisinin gelişmesi ile nöro donanımlar uygulamaya özel olarak, FPGA yongaları üzerinde tasarlanmaya başlanmıştır. FPGA donanımları gerçek zamanlı uygulamalar için gereken hesaplama hızını, esnek mimarileri ile daha ucuz donanımlarla oluşturmayı mümkün kılmıştır [21, 147].

6.3. Nümerik Tanımlama

FPGA üzerinde YSA geliştirme uygulamalarında donanım maliyetini etkileyen bir diğer önemli parametre ise tasarımın kullandığı nümerik tanımlama yöntemidir. Aşağıda sayısal hesaplamalarda kullanılan günümüz modern nümerik tanımlama metotlarının oluşması süreci ve bu tez çalışmasında kullanılan iki farklı sayı sistemi hakkında bilgiler verilmiştir.

Rakamların insanlık tarihinde kullanılması 4000 yıllık bir geçmişe sahiptir. Günlük hayatımızda kullandığımız desimal (onluk) sayı sistemi, MS 500 yıllarında insan el ve ayaklarında bulunan 10 parmaktan esinlenilerek Hindular tarafından bulunmuştur [148].

AHMET TURAN ÖZDEMİR - ahmetozdemir.yes.edu.tr

Roma rakamlarının matematiksel yorum zorluğu ve işlevsel zayıflıkları, İlk defa Hindu sayı sisteminde bir sayı olarak ifade edilen "sıfırın" tanımı ile aşılmıştır. Dokuzuncu yüzyılda Araplar tarafından geliştirilen bu sistem, sıfırın avantajını kullanmak amacıyla ancak 16. yüzyılda Avrupalılar tarafından kullanılmaya başlanmıştır [149]. Avrupa'da Arap sayıları olarak ifade edilen desimal (ondalıklı) sayılar, Araplar tarafından Hindu sayıları olarak ifade edilir. Bu iki kültürü de tanımak amacıyla, desimal sayılar genellikle Hindu-Arap sayıları olarak ifade edilir [148, 150, 151].

Günümüz bilgisayar mimarilerinde kullanılan modern hesaplama sisteminin temeli olan ikili (binary) sayı sistemi, ilk defa dördüncü yüzyılda şiirlerindeki uzun heceleri matematiksel biçimde kısaltmak amacı ile Hindu şair ve yazar Pingala tarafından geliştirilmiştir [151]. İkili sayı sistemi 1500 yıl aradan sonra ilk defa, 1600'lü yıllarda deneyciliğin babası olarak bilinen İngiliz bilim adamı Francis Bacon tarafından, ikili kodlama (binary encoding) teorisi ile yeniden yorumlanmıştır. 1854 yılında İngiliz matematikçi George Boole, Boole Cebri (Boolean Algebra) olarak bilinen ikili aritmetiği, çalışması ile tanımlamıştır. 1937 yılında bilişim teorisinin babası olarak bilinen Amerikalı elektronik mühendisi Claude Shannon, elektronik devrelerde kullanılan kapı (röle) sayılarının, ikili sayı sistemi ve Boole Cebri kullanılarak nasıl azaltılacağını tarif etmiş ve bu çalışması ile sayısal bilgisayar ve sayısal devrelerin temellerini atmıştır.

Sayısal sistemlerin ortaya çıkması ile bu mimarilerde kullanılmak üzere, sayısal hesaplama teknikleri ve nümerik tanımlamalar ortaya çıkmıştır. Modern bilgisayar mimarilerinde kullanılan sayı formatları ve tarihsel gelişim süreci aşağıda kısaca özetlenmiştir.

En eski bilgisayardan üçü ENIAC [152], UNIVAC [153] ve IBM 650 [154] aritmetik işlemleri desimal olarak yani onluk sayı sisteminde gerçekleştirirken [155], yine aynı dönemde EDSAC [156], EDVAC [157] ve ORDVAC [158] bilgisayarları aritmetik işlemleri binary olarak yani ikili sistemde gerçekleştirmekteydiler. Bilgisayarların yapı taşı olan iki durumlu transistörlerin binary sayı sistemindeki avantajına rağmen, bazı bilgisayar üreticileri dört bit ikili kodlanmış onlu (BCD, Binary Coded Decimal) [159] sayı sistemini kullanmaya devam ettiler. Zamanla uygulamalarda ihtiyaç duyulan yüksek hızlı hesaplama, binary aritmetiğin kullanımını gerektirmiş ve binary aritmetiği bir hesaplama standardı haline gelmiştir [151].

AHMET TURAL ÖZDEMİR, atural@ceng.ege.edu.tr

Sabit noktalı nümerik tanımlamanın aksine, bilgisayar üreticileri kendi makineleri için kendilerine özgü kayan noktalı nümerik tanımlama standartları geliştirmişlerdir [160]. Bunun sonucu olarak farklı bilgisayarlarda kullanılan farklı kayan noktalı nümerik tanımlama algoritmaları yüzünden, aynı veri farklı sonuçlar ile yorumlanmaktaydı. Bu yüzden ortak bir kayan noktalı nümerik tanımlama standardına ihtiyaç duyulmuştur. 1985 yılında IEEE tarafından 754 numaralı referans ile tanımlanan standart ile kayan noktalı nümerik tanımlama yoğun bir şekilde kullanım alanı bulmuş [161] ve bugün revize edilmiş hali olan IEEE 754 2008 standardı ile son haline getirilmiştir [162]. Bütün bu süreç içerisinde bilgisayar programlama dilleri de sabit noktalı nümerik tanımlamadan, kayan noktalı nümerik tanımlamaya doğru evrimleşmiştir [151].

6.3.1. Sabit Noktalı Nümerik Tanımlama

Sabit noktalı nümerik tanımlama tamsayı (integer) sayı formatına benzemektedir. Ancak sabit noktalı nümerik tanımlamada, kesiri belirleyen radix noktası integer sayı formatında olduğu gibi sağdaki en düşük değerli bitin (LSB, Least Significant Bit) hemen sağında yer almayabilir. Radix noktasının yeri bellidir ve bu noktanın sağındaki solundaki bit sayıları sabittir. Sabit noktalı nümerik tanımlamada aritmetik işlemlerde kullanılan bütün operantlarda radix noktasının yeri aynıdır. Radix noktasının sağında kalan bitler fraction (*F*) olarak ifade edilir ve küsuratı ifade ederler. Bu noktanın solunda yer alan bitler ise integer (*I*) değeri ifade ederler. Sabit noktalı nümerik tanımlamada noktanın sağındaki ve solundaki bitler yani veri uzunluğu, hassasiyeti belirler. Sabit noktalı nümerik tanımlamada kullanılan bit sayısı, tanım aralığını ve hassasiyeti belirler. Fakat bit sayısı sınırlı olduğu için tasarımcı aritmetik işlemler sonucunda oluşacak nümerik kayıpları ve taşmaları göz önünde bulundurmalıdır.

Bu tez çalışmasında soldaki en yüksek değerlikli bit (MSB, Most Significant Bit) işaret biti, sonrasındaki yedi bit integer (tamsayı) ve kalan sekiz bit fraction (kesir) olmak üzere, 16 bit sabit noktalı nümerik sayı formatı kullanılmıştır. 64,125 değeri için binary ifade Şekil 6.11'de verilmiştir.

$\pm$	2^6	2^5	2^4	2^3	2^2	2^1	2^0		2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}
0	1	0	0	0	0	0	0	.	0	0	1	0	0	0	0	0
1	2	3	4	5	6	7	8		9	10	11	12	13	14	15	16
MSB				***I***				**+**				***F***				***LSB***

Şekil 6.11. 16 bit sabit noktalı nümerik sayı formatı.

AHMET TURAN ÖZDEMİR, aturan@erciyes.edu.tr

16 bit nümerik tanımlama ile yaklaşık ±128 genişliğindeki nümerik aralık, $p = 2^{-8}$ nümerik hassasiyet ile ifade edilebilmektedir. Radix noktası sağı ve solundaki bitleri eşit olarak sekizer bite bölmüştür ve bütün hesaplamalarda bu noktanın yeri sabittir. Nokta bit dizisi içerisinde görüntülenmez, tasarımcı noktanın yerini bilerek işlemleri yapar. Bütün aritmetik işlemler temel binary aritmetiği mantığı ile gerçekleştirilir. Çarpım işlemleri sonucunda, sonuç bit dizisi iki katına çıkar. Yani 16x16 çarpım sonucunda 32 bitlik bir cevap oluşur. Bu değer tekrar 16 bit ile ifade edildiğinde veride taşma oluyorsa sonuç hatalı çıkacaktır. Bu tez çalışmasında kullanılan veri seti ve ağırlıklar taşmalara sebep olmayacak şekilde seçilmiştir.

VHDL kütüphanesi işaret bitli sabit noktalı nümerik sayı formatını desteklemektedir. MSB'den yani işaret bitinden sonraki bitlere binary aritmetiği uygulanır. İşlem sonucunda elde edilen 15 bitlik veri için işaret biti atanır ve sonuç tekrar 16 bit ile ifade edilir.

Sayısal sistemlerde en kısa işlem süreleri sabit noktalı nümerik tanımlama ile yakalanmaktadır. Sabit noktalı nümerik tanımlama binary aritmetiğini kullandığı için donanım seviyesine en yakın olan tanımlama tekniğidir. Bu yapısal sadeliği yüzünden donanım kaynağı ihtiyacı da oldukça düşüktür. Zaten sabit noktalı nümerik tanımlamanın hızlı cevap süresinde en belirleyici katkı, verinin donanım içerisinde maruz kaldığı propagasyonun en az olmasından ileri gelmektedir. Az sayıdaki donanım kaynağı, veri akışındaki gecikmeleri minimize eder.

6.3.2. Kayan Noktalı Nümerik Tanımlama

IEEE'nin 754 numaralı standardı ile ilk defa 1985 yılında ortaya konulmuş olan kayan noktalı nümerik tanımlama standardı (IEEE 754 Floating Point Numerical Description) [161], binary sayı sisteminin ticari olarak yeni bir formatta yorumlanmasını sağlamıştır. Bugün artık IEEE 754 2008 [162] ile son haline getirilmiş olan bu nümerik format tarih boyunca en çok kullanılan nümerik standart olmuştur.

Kayan noktalı nümerik tanımlama en temel şekli olan 1985'te yayınlanan ilk standarda göre dört ana formatta, temel ve genişletilmiş olmak üzere iki grup altında ifade edilir. Bu dört kayan noktalı nümerik tanımlama formatı aşağıdaki gibi ifade edilmektedir.

AHMET TURAN ÖZDEMİR - atozdemir@erciyes.edu.tr

Temel

- 32 bit kayan noktalı nümerik tanımlama, tek hassasiyetli (single precision)
- 64 bit kayan noktalı nümerik tanımlama, çift hassasiyetli (double precision)

Genişletilmiş

- 43 veya daha fazla sayıda bit ile ifade edilen kayan noktalı nümerik tanımlama (single-extendend precision)
- 79 veya daha fazla sayıda bit ile ifade edilen kayan noktalı nümerik tanımlama (double-extendend precision)

IEEE 754 2008 ile 16 bit hassasiyetten (half precision) 128 bit hassasiyete (Quadruple precision) kadar ifade edilebilen kayan noktalı nümerik tanımlama formatları geliştirilmiştir. Bununla birlikte bu formatlar 32 ve 64 bit kayan noktalı nümerik formatlar kadar geniş bir kullanım alanı bulamamışlardır.

Kayan noktalı nümerik tanımlama standardıyla sayıların değişik seçeneklerle (en yakın değere, pozitif sonsuza doğru, negatif sonsuza doğru ve sıfır etrafında) yuvarlanması, pozitif ve negatif yönlü taşmalar, sonsuz ve işaret bitli sıfır gibi özel sayılar tanımlanmıştır. Toplama, çıkarma, çarpma ve bölme gibi aritmetik işlemler de bu standart içerisinde tanımlanmıştır. Kayan noktalı nümerik tanımlama VHDL kütüphanesi tarafından da desteklenmektedir.

Kayan noktalı nümerik sayı formatı üç ana bileşene bölünebilir. Bunlar işaret biti (ib), mantissa (sabit noktalı nümerik formattaki fraction yani kesire karşılık gelmektedir) ve exponent (tamsayı) bileşenleridir. 32 bit nümerik tanımlama için kullanılan sayı formatı Şekil 6.12'de verilmiştir. 0 ile 22 arasındaki toplam 23 bit mantissa'yı, 23 ile 30 arasındaki 8 bit exponent'i ve en soldaki bit işaret bitini temsil etmektedir. İşaret biti sabit noktalı nümerik tanımlama formatında olduğu gibi 0 ise pozitif, 1 ise negatif değer tanımı yapar.

ib	Exponent (8 bit)			Mantissa (23 bit)		
31	30	…..……	23	22	……	0

Şekil 6.12. 32 bit kayan noktalı nümerik sayı formatı.

32 bit kayan noktalı nümerik sayı formatı $\pm 2^{-126}$ ile $2^{127}(2-2^{-23})$ aralığını yani desimal olarak yaklaşık $\pm 1{,}8\times 10^{-38}$ ile $3{,}4\times 10^{38}$ aralığını tanımlar. Binary radix noktasının yeri sabit değildir ve bu yüzden kayan noktalı tanımlama adını alır. 32 bit nümerik sayı formatı

AHMET TURAN ÖZDEMİR - aturan@erciyes.edu.tr

aritmetik işlemler için binary formata göre daha büyük hafıza alanları işgal eder ve işlemleri daha uzun sürelerde tamamlar. Fakat mantissa'dan görüleceği gibi çok yüksek bir hassasiyet ile cevap üretir. Birçok yazılım veya uygulama için nümerik veri hassasiyeti büyük önem taşımaktadır. Kayan noktalı nümerik tanımlama ile tasarımcıya istediği kadar hassas nümerik işlem olanağı sunulmaktadır.

6.3.3. Sabit ve Kayan Noktalı Nümerik Tanımlamaların Karşılaştırılması

Sabit noktalı nümerik tanımlama, kayan noktalı nümerik tanımlamaya göre sınırlı kabiliyetlere sahiptir. Buna rağmen geçmişteki yonga teknolojilerinin sahip olduğu sınırlı tümleşik yoğunluktan dolayı sayısal sistemlerde en uzun süre kullanılmış olan sayı formatıdır. Bununla birlikte bazı yeni nesil teknolojiler, sonsuza yakınsayan büyük değerli veya sıfıra yakınsayan küçük değerli sayılar kullanmaktadırlar. Fakat bazı uygulamalarda nümerik aralık kadar hassasiyet de önemlidir. Unutulmamalıdır ki 0 ile 1 arasında sonsuz sayıda nümerik değer bulunmaktadır. Kayan noktalı nümerik tanımlama çok büyük bir nümerik aralığı çok yüksek derecede hassasiyet ile ifade etmektedir.

Çok yüksek bir oranda doğruluk sağlaması sebebiyle; kayan noktalı nümerik format en ideal nümerik tanımlama yöntemidir. Bu özelliği sayesinde uygulamalarda en yaygın kullanılan çözümdür. Bu, neden kayan noktalı tanımlamanın genel amaçlı hesaplamalarda sıklıkla kullanıldığını ve birçok sayısal entegre üreticisinin özel kütüphaneler ile destek sunduğunu açıklamaktadır. Fakat FPGA yongalarındaki sınırlı kaynaklar sebebiyle nümerik tanımlamalar için kayan noktalı nümerik tanımlamanın kullanımı, sabit noktalı nümerik tanımlamada olduğu gibi daha etkili kaynak kullanan tanımlama yöntemlerinin kullanımı kadar uygun değildir [163].

Kayan noktalı nümerik tanımlamanın kullanıldığı donanımlar, sabit noktalıya ilave lojik kapılardan meydana gelir. 32 bit kayan noktalı çarpıcının kullandığı alan, 16 bit sabit-noktalı çarpıcının kullandığından çok daha fazladır. Aynı durum doğal olarak aktivasyon fonksiyonu, toplama ve çıkarma birimleri için de geçerlidir. FPGA tabanlı YSA'larda kayan noktalı formatın ihtiyaç duyduğu daha fazla kaynak ve alan ihtiyacı, ancak tasarlanan YSA boyutunun küçültülmesi ile FPGA gerçeklemesini mümkün kılar.

K.R. Nichols, M.A. Moussa ve S.M. Areibi FPGA tabanlı YSA yapılarını, kaynak kullanımları, optimum değerler vs. cinsinden test etmişlerdir [164]. BP öğrenme algoritması kullanılarak oluşturulan YSA ile XOR problemini aşağıda verilen mimariler ile çözmüşlerdir;

AHMET TURAN ÖZDEMİR - aturan@erciyes.edu.tr

i) 32 bit kayan noktalı nümerik tanımlama ile gerçekleştirilmiş FPGA tabanlı YSA,
ii) 16 bit sabit noktalı nümerik tanımlama ile gerçekleştirilmiş FPGA tabanlı YSA,
iii) Yazılım tabanlı YSA.

Klasik XOR probleminin çözümünde kullanılan bir YSA için mikroişlemci ve FPGA performansları Tablo 6.2'de görülmektedir. Buradan hassasiyetin, oluşan yoğunluk ve kullanılan kaynak sayısı üzerinde direk etkisi olduğu anlaşılır. Günümüz FPGA'ları göz önüne alınacak olursa düşük nümerik hassasiyete sahip olan 16 bit sabit noktalı nümerik tanımlama formatı en hızlı donanım cevabını üretmektedir.

Tablo 6.2. Mantıksal XOR problemini çözmekte kullanılan yapılar ve sonuçlar [164].

XOR YSA Mimarisi	**Duyarlık (Hassasiyet)**	**Toplam Alan CLB [Slices]**	**Maksimum Clock (MHz)**	**Saniyede Değişen Ağırlık (WUPS)**
Xilinx Virtex Excv2000e	16 bit Sabit Noktalı	1.239 [2.478]	10	62.762
Xilinx Viretx II Xc2v8000	32 bit Kayan Noktalı	8.334,75 [33.339]	1,25	5.172
Intel Pentium III	32 bit Kayan Noktalı	Sonsuz sayıda	733	1.467

Tablo 6.2'de görüldüğü gibi nümerik tanımlamada kullanılan bit sayısı azaldıkça donanım hızı artmaktadır ve kullanılan kaynak sayısı azalmaktadır. 10 MHz sistem clock girişi olan bir FPGA üzerinde 16 bit nümerik tanımlama ile oluşturulmuş YSA mimarisi saniyede 62.762 kez değişim gösterirken, aynı problemin çözümlendiği 300 MHz sistem clock girişi olan bir Pentium III işlemcide saniyede 1.467 değişim görünmektedir. 10 MHz clock girişi olan FPGA, 700 MHz ile çalışan güçlü bir işlemciden daha hızlı çalışmaktadır. Bunun sebebi FPGA'ların sahip olduğu paralel işlem yeteneğidir. FPGA üzerinde aynı yapı kayan noktalı tanımla kullanılarak tasarlandığında, değişimin 5.172'ye kadar düştüğü ve kullanılan kaynak sayısının ise arttığı görülmektedir. Bununla birlikte 32 bit nümerik tanımlamanın ağın nümerik hassasiyetini arttırdığı unutulmamalıdır. Bu bakımdan kaynak kullanımı ile nümerik tanımlama arasında bir ilişki vardır ve bir problemin çözümünde problemin ihtiyaç duyduğu kritere göre, doğruluk veya alan ölçütleri gözetilerek ağ tasarlanmalıdır.

AHMET TURAN ÖZDEMİR, aturan@erciyes.edu.tr

6.4. 8x2x1 Boyutlu YSA'nın FPGA Üzerinde Gerçeklenmesi

Bu çalışmada donanım geliştirme platformu olarak Aletra Cyclone III EP3C120F780 yongası 50 MHz sistem clock frekansı ile kullanılmıştır. Cyclone FPGA ailesi gelişmiş fonksiyonlara sahip iç donanımları ile düşük güç tüketimi ve düşük maliyet avantajları sunar. EP3C120, Cyclone III ailesinin en büyük FPGA yongalarından biridir. Cyclone III ailesi düşük güç tüketimi ve maliyetine karşın yoğun bir donanım kaynağına sahiptir. Bu sayede tasarımcı büyük ölçekli devreleri tek bir FPGA yongası içerisinde oluşturabilir. Bu tez çalışmasında ailenin en büyük yongalarından biri olan EP3C120'nin kullanılmasının sebebi, donanım modeli oluşturulurken FPGA yongasından kaynaklı donanım sınırlandırmalarının yaşanmamasıdır. Son hali verilen bir sayısal tasarım, aynı aile içerisinden ihtiyacı karşılayabilecek daha küçük bir FPGA üzerinde kolayca gerçekleştirilebilir.

Paralel çalışan donanımlar genellikle giriş ve çıkışlarda her bir biti temsil edecek bir I/O terminali ve matematiksel ifadelerin ayrı ayrı fakat aynı zamanda koşturulduğu bağımsız matematiksel operatörlere ihtiyaç duyarlar. Cyclone III ailesine ait bazı yongaların sahip oldukları donanım kaynakları Tablo 6.3'te verilmiştir. EP3C120 18x18 bit genişlikli 288 adet gömülü çarpıcıya sahiptir, arzu edilirse gömülü çarpıcılar iki katı yani 576 adet 9x9 bit genişlikli çarpıcı olarak da kullanılabilir. Ayrıca gömülü çarpıcılara ilave olarak, EP3C120'nin sahip olduğu diğer donanım kaynaklarını kullanarak yapılandırılan, maksimum 16x16 bit genişlikli 432 adet yazılımsal çarpıcı daha oluşturulabilir. Tablodaki I/O sayısı maksimum değer olarak verilmiştir ve bu sayı EP3C120 için 531'dir. EP3C120 serisi içerisinde terminal sayıları ve yonga kılıf yapıları farklı olan FPGA yongaları bulunmakla birlikte, hepsi aynı sayıda donanım kaynağına sahiptir. Bu çalışmada kullanılan EP3C120 serisi FPGA yongasının genişletilmiş ismi Cyclone III EP3C120F780'dir. Sondaki F780 tanımlaması ise bu FPGA'nın kılıf tipini ifade eden bir belirteçtir. F780, 2.6 mm yüksekliğinde 29x29 mm boyutunda kare bir plastik kılıfı tanımlamaktadır. Bir FPGA yongasının sipariş kodunda bu genişletilmiş tanımlama ismin sonunda yer alan ve üç karakterden oluşan sırası ile çalışma sıcaklık aralığı, hız ve isteğe bağlı tanımlamaların bulunduğu bir son ek vardır. İsteğe bağlı tanımlamaların yapıldığı son karakter ise entegrenin ürün numunesi veya kurşunsuz olduğunu tanımlamak için verilebilir. Cyclone III Device Handbook isimli dokümanda bu ailenin bütün özellikleri detaylı olarak verilmiştir [119].

AHMET TURAN ÖZDEMİR - atozdemir@erciyes.edu.tr

Tablo 6.3. Cyclone III ailesinden bazı FPGA'ların donanım kaynak özellikleri.

Aile	Yonga	Lojik Eleman	Toplam RAM (bit)	18x18 Çarpıcılar	PLL Sayısı	I/O Sayısı
Cyclone III	EP3C5	5.136	423.936	23	2	182
	EP3C25	24.624	608.256	66	4	215
	EP3C55	55.859	2.396.160	156	4	377
	EP3C80	81.264	2.810.880	244	4	429
	EP3C120	**119.088**	**3.981.312**	**288**	**4**	**531**

Cyclone III ailesi taşınabilir (mobil) uygulamalar için yüksek yoğunluklu hafıza, gömülü çarpıcı ve I/O seçenekleri sunar. Cyclone III birçok dış hafıza erişim ve I/O iletişim protokolünü destekler. Bu aile için özel fonksiyonları icra etmek üretilmiş çok sayıda uyarlanabilir Hazır Tasarım (IP, Intellectual Property) çekirdeği bulunmaktadır. Bu IP çekirdekleri birkaç küçük uyarlama ile tasarımların içerisine eklenerek kullanılabilir [119].

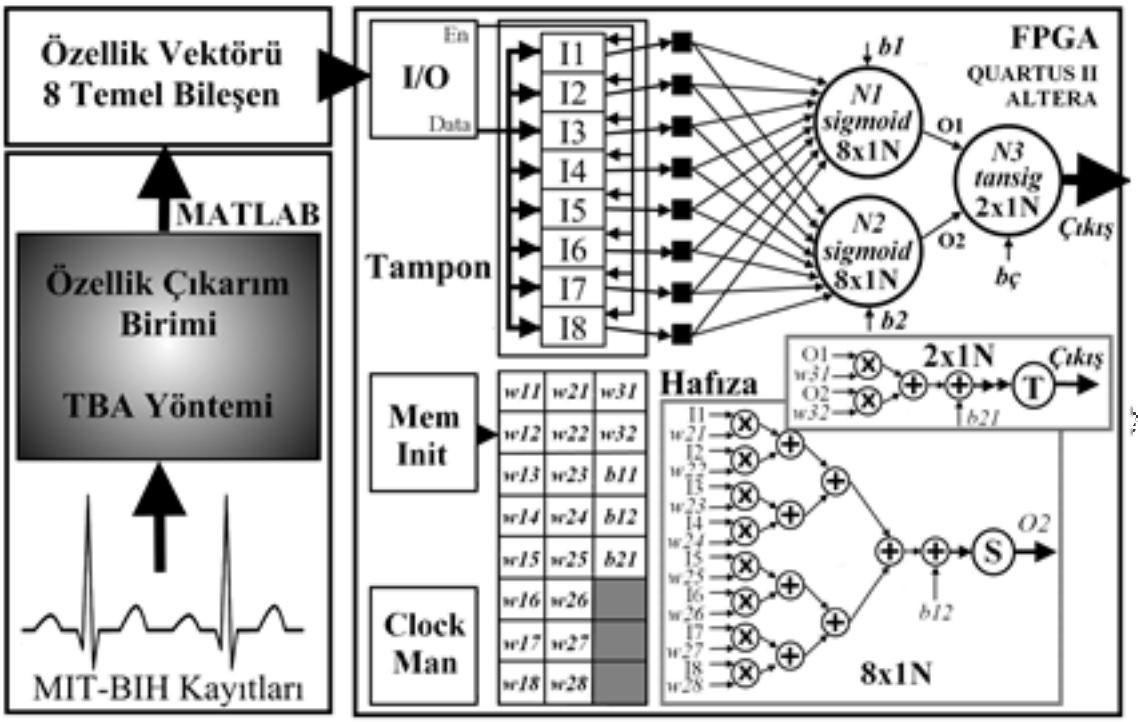
Bu tez çalışmasında Cyclone III EP3C120F780 FPGA yongası üzerinde, IEEE 754 32 bit kayan noktalı ve 16 bit sabit noktalı nümerik tanımlamalar kullanılarak, 8x2x1 boyutlu YSA tabanlı bir otomatik PVC aritmi sınıflandırıcı %97,66 ve %96,54 doğruluklarla tasarlanmıştır. Bu yapılar sırası ile FPANN ve FXANN olarak isimlendirilir. FPANN IEEE 754 32 bit kayan noktalı nümerik tanımlama kullanılarak FPGA üzerinde oluşturulmuş 8x2x1 boyutlu YSA modelinin donanım adıdır. FXANN ise aynı boyuttaki bu YSA modelinin 16 bit sabit noktalı nümerik tanımlama kullanılarak FPGA üzerinde donanım olarak tasarlanmış adıdır. FPANN donanımı gizli katmanında gerçek sigmoid aktivasyon fonksiyonu kullanılarak tasarlanmış, FXANN ise gizi katmanında parçalı doğrusal sigmoid aktivasyon fonksiyonu kullanılarak tasarlanmıştır. Hatırlanacağı üzere, FPANN ağ parametreleri 5. Bölüm'ün sonunda verilmişti, FXANN modelinin ağ parametreleri ise bu bölüm içerisinde, parçalı doğrusal sigmoid fonksiyonu ve bu fonksiyonun Matlab uyarlaması anlatıldıktan sonra verilecektir. 16 bit nümerik tanımlamanın ve parçalı doğrusal sigmoid aktivasyon fonksiyonunun kullanıldığı FXANN yapısı ağın hatasını arttırmış ancak donanım maliyeti ve cevap süresini azaltmıştır. Bu iki yapının kaynak kullanımları, donanım gerçeklemeleri ve performans ölçütleri bu bölümün sonunda detaylı olarak verilecektir.

FPANN ve FXANN birbirlerinden farklı özelliklere sahip donanımlar olmalarına karşın her ikisi de 8x2x1 boyutunda olup fonksiyonel olarak özdeş birimlerden oluşmaktadırlar. 8x2x1 boyutlu YSA'nın donanımsal olarak blok diyagramı ve donanımın Matlab ile bağlantısı Şekil

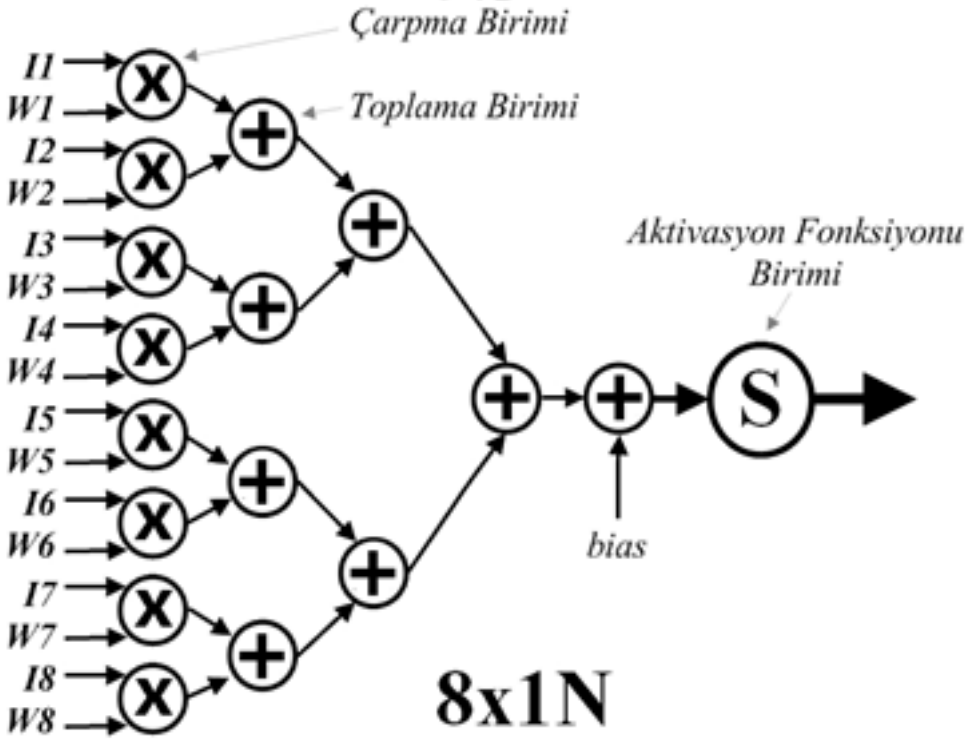
AHMET TURAN DEMİR - turandemir@erciyes.edu.tr

6.13'te verilmiştir. FPGA içerisindeki YSA donanımı şekilden görüldüğü gibi sekiz ana birim ve bunların çok sayıdaki alt birimlerinden meydana gelir. Bu birimleri kısaca açıklamak gerekirse, ClkMan birimi FPGA içerisindeki clock organizasyonunu sağlayan birimdir. Verinin yayılım yoluyla işlendiği paralel mimaride, her bir işlem birimi için bir önceki verinin hesaplaması bitene kadar pasif tutulur. Ayrıca işlemini tamamlayan her bir birimin çıkışı hafızaya alınarak, bu birimler de pasif duruma geçirilir. Ardışıl işlem bloklarının ne zaman ve ne kadar süreler ile çalışacakları ClkMan birimi tarafından kontrol edilir. Ancak bunun yapılabilmesi için hangi birimin ne sürede işlemini bitirdiğinin bilinmesi gerekmektedir. ClkMan birimi yanlış hesaplanmış verilerin oluşmasını önlemekte, sistem güç tüketimini düşürmekte ve de YSA'nın her girişini aynı süreler ile kontrol ettiği için verinin ağ üzerinde bütün girişler boyunca tümüyle paralel olarak yayılmasını sağlamaktadır. MemInit birimi Matlab'den elde edilen ağırlık ve biasların saklandığı gömülü hafıza birimidir. Eğitimi Matlab ortamında tamamlanan ağların, ağırlık ve bias değerleri bu hafızadan çağrılarak donanım cevapları oluşturulur. FPGA yongası ilk enerjilendiğinde, işlemci elemanlar kendi ağırlık ve bias değerleri ile güncellenir. I/O bloğu MLP YSA donanımının veri yoludur ve ağın girişidir. Tampon birimi veri yolu üzerinden aktarılan seri giriş bilgisinden sekiz girişi yakalayarak ayıklar ve bu girişler ile gizli katmandaki işlemci elemanları besler. 8x1 Nöron bloğu (8x1N) tasarımın en büyük alan kaplayan birimidir ve içerisinde çok sayıda çarpıcı, toplayıcı gibi alt hesaplama birimleri barındırır. 8x1N birimi içerisindeki sigmoid aktivasyon fonksiyonu birimi ise en çok kaynak tüketimine sahip olan alt birimdir ve Şekil 6.14'te blok diyagram olarak gösterilmiştir. 2x1 Nöron (2x1N) birimi çıkış işlemci elemanıdır ve purelin aktivasyon fonksiyonuna sahiptir. FPANN için I/O birimi veri yolu 32 bit, FXANN için 16 bittir. Şekil 6.13, 8x2x1 boyutlu YSA donanımı için genel bir gösterimdir. Birçok yönden farklılıklara sahip olan bu donanımların ayrıntıları ileride verilecektir.

AHMET TORAY DEMİR ahmet@ceyer.com

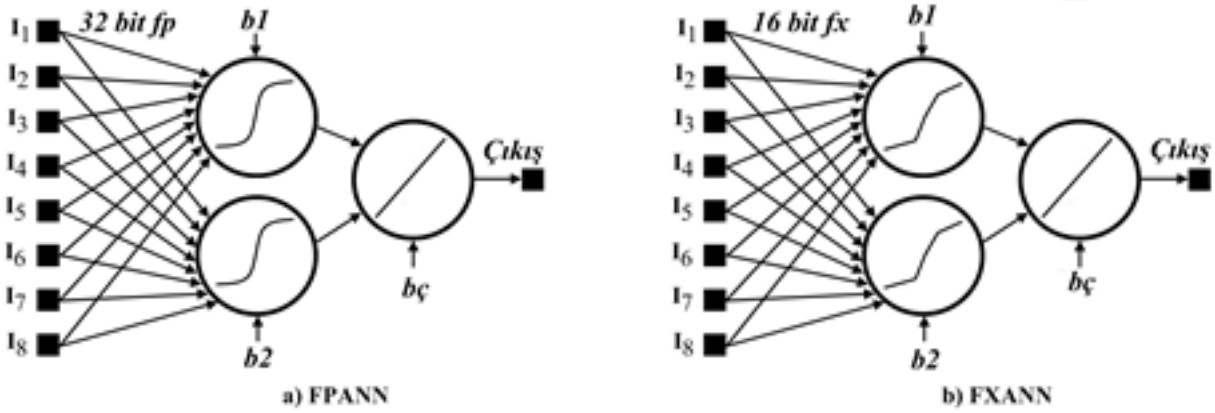


Şekil 6.13. EKG kayıtlarının FPGA modeline uygulanması.



Şekil 6.14. Sekiz girişli bir işlemci elemanın blok diyagramı.

FPANN ve FXANN için aktivasyon fonksiyonu ve veri yolu büyüklüğünü ifade eden bir gösterim Şekil 6.15'te verilmiştir. FPANN'de bütün ağ boyunca 32 bit kayan noktalı nümerik format ile işlem yapılır. FXANN ise girişlerine uygulanan 16 bit sabit noktalı nümerik formatı kullanır. Fakat çarpım terimlerinden sonra 32 bit sabit noktalı nümerik değerler oluşur. Bu değerler çarpıcıların çıkışlarında bulunan bir birim tarafından tekrar 16 bit sabit noktalı nümerik formata çevrilir. FXANN içerisindeki 32 bitlik değerlerin 16 bite dönüştürülmesinin sebebi, 16 bitlik donanımların 32 bitlik donanımlara göre daha az kaynak kullanmasıdır.



Şekil 6.15. a) FPANN b) FXANN modelleri.

8x2x1 boyutlu YSA modelleri paralel bir lojik mimari ile FPGA içerisinde tasarlanırken Altera Quartus II yazılımı kullanılmıştır. Tasarım VHDL kodları, şematik dosyalar ve MegaWizard kütüphanesinden alınan bazı IP çekirdekleri kullanılarak karma bir model halinde tasarlanmıştır. Quartus II ile tasarım yapılırken, tasarımda kullanılan dosyalardan birinin (bu dosya VHDL veya şematik olabilir) "top level entity" yani ana dosya olarak tanımlanması gerekmektedir. Bu dosya, alt birimleri ve bu birimlerin birbirleri ile olan bağlantılarını tanımlar. Bu tanımlamalar giriş çıkış terminallerinin bit genişlikleri ve içeride kullanılan sinyallerin isim ve bit genişlikleridir. Her bir alt birim yine VHDL veya şematik dosyası olabilir ve bu dosyalar özel birer isim ile kaydedilir. Tasarım akışı FPANN ve FXANN için aynı olacaktır, devam eden alt bölümde FPANN için yapılan tasarım detaylandırılmıştır.

6.4.1. IEEE 754 32 Bit Kayan Noktalı Nümerik Tanımlama ile FPGA Üzerinde YSA Donanımı Gerçekleştirilmesi (FPANN)

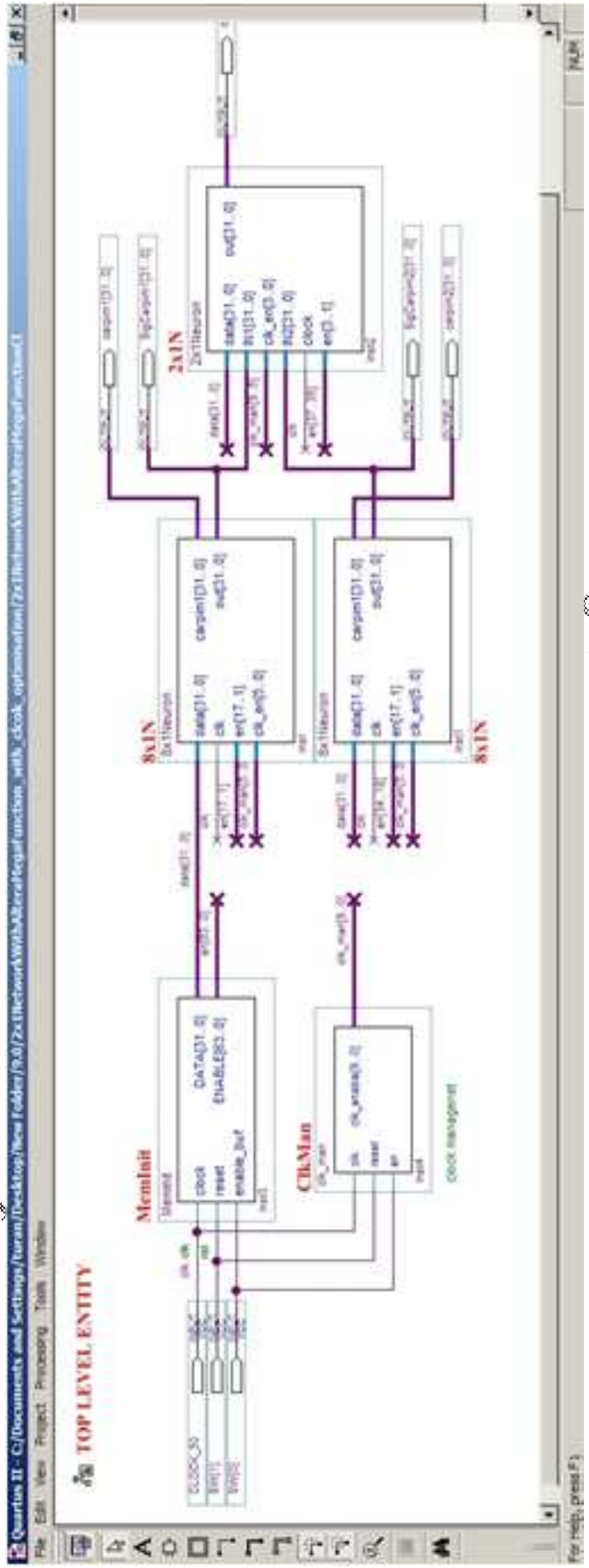
FPANN Denklem 6.1 ile verilen doğrusal olmayan sigmoid aktivasyon fonksiyonunu kullanmaktadır. Burada net_j gizli katmandaki işlemci elemanın sinaptik toplamını, O_j ise işlemci elemanın çıkışını vermektedir. Bir başka ifade ile O_j aktivasyon fonksiyonu bloğunun çıkışıdır.

$$O_j = \frac{1}{1+e^{-net_j}} \quad (6.1)$$

FPANN, -2^{-126} ile $+(2-2^{-23})\times 2^{127}$ arasında değişen bir nümerik tanımlama aralığına sahip 32 bit kayan noktalı nümerik tanımlamayı kullanmaktadır. Bu nümerik formata "single precission floating point number" sistem de denilmektedir. FPAAN donanım gerçeklemesinde en zor kısım sigmoid alt bloğunun oluşturulmasıdır. FPGA üzerinde donanım modellenirken, uygun FPGA yongasının seçiminde en büyük ölçüt tasarımın ihtiyaç duyduğu donanım kaynaklarıdır. 32 bit kayan noktalı nümerik format aritmetik işlemlerin gerçekleştirilmesi için Altera FPGA yongalarındaki gömülü ve yazılımsal çarpıcıların çok sayıda kullanılması gerekecektir. Altera bu çarpıcıları DSP eleman olarak isimlendirmektedir [95, 119, 161, 164].

Altera Quartus II programı kullanılarak tasarlanmış olan FPANN'nin ana tasarım dosyası (top level entity) Şekil 6.16'da verilmiştir. Burada her bir ana birim kolaylıkla görülebilmektedir. Ayrıca FPANN tasarımının dosya organizasyonu, her bir birimin kaynak kullanımı ve bütün tasarım süreçlerinin onaylanmış olduğuna dair bilgilerin bulunduğu tasarım ekranı görüntüsü Şekil 6.17'de verilmiştir. Bu ekran görüntüsünde tasarımda kullanılan toplam lojik eleman sayısı, toplam hafıza elemanı sayısı ve toplam çarpıcı elemanların sayısı bütün tasarım ve her bir blok için ayrı ayrı verilmiştir.

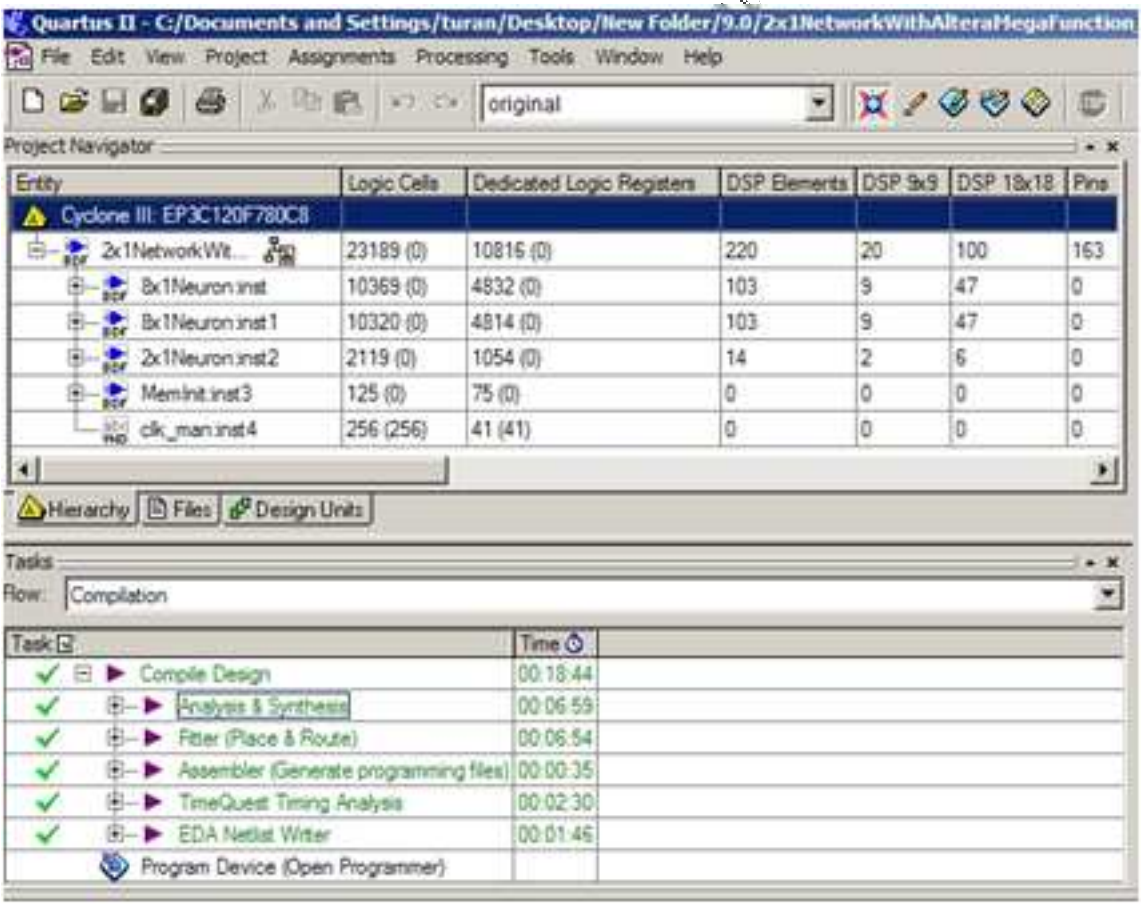
AHMET TURAN ÖZDEMİR, aturan@erciyes.edu.tr



Şekil 6.16. FPANN ana tasarım dosyası.

AHMET ÇIÇIYES.edu.tr

FPANN, 94'ü gizli katman işlemci elemanları içerisindeki iki adet sigmoid aktivasyon fonksiyonu bloğu ve 126'sı sistemde yer alan 18 çarpım elemanı tarafından kullanılan toplam 220 adet 9x9 (9 bitlik çarpıcı) gömülü çarpıcı birimine ihtiyaç duyar. 9x9 gömülü çarpıcıların her biri DSP eleman olarak isimlendirilir. 220 adet DSP elemanın 100 tanesi 18x18 (18 bitlik çarpıcı), 20 tanesi ise 9x9 bit boyutlu çarpıcılardan meydana gelmektedir. Her bir 8x1N birimi 103 adet DSP eleman kullanır ancak bu kaynağın 47 adet DSP kullanımı tek başına sigmoid aktivasyon fonksiyonu bloğu tarafından yapılmaktadır. Bu haliyle sigmoid aktivasyon fonksiyonu bloğu tasarımın en pahalı birimidir. İkinci en çok DSP işlemci elemanı kullanan birim ise çarpma işlemi birimidir. Her bir 32 bit kayan noktalı çarpma işlemi için 7 adet DSP işlemci eleman kullanılır. Toplayıcılar DSP işlemci elemanlarına ihtiyaç duymazlar. Fakat 32 bit kayan noktalı nümerik değerlerin toplamı için yine de çok sayıda lojik elemana ihtiyaç duyulmaktadır. FPANN ve içerisindeki dört ana bloğun donanım kaynak kullanımları toplam lojik eleman, toplam DSP işlemci elemanı ve hafıza kaynakları cinsinden, adet ve toplam kullanımın yüzdeleri cinsinden Tablo 6.4'te ayrıntılı olarak verilmiştir.



Şekil 6.17. Kaynak kullanımları ve tasarım süreçlerinin onayı (FPANN).

Tablo 6.4. FPANN ve ana bloklarının donanım kaynak kullanımları.

	2* 8x1N	2x1N	MemInit	ClockMan	FPANN
Toplam Lojik Eleman	20.689 89%	2.119 9%	125 ≤ 1%	256 ≤ 1%	23.189 100%
Toplam DSP Eleman	206 94%	14 6%	0	0	220 100%
Toplam Hafıza Elemanı	9.646 89%	1.054 10%	75 ≤ 1%	41 ≤ 1%	10.816 100%

**FPANN iki adet 8x1 bloğuna sahiptir*

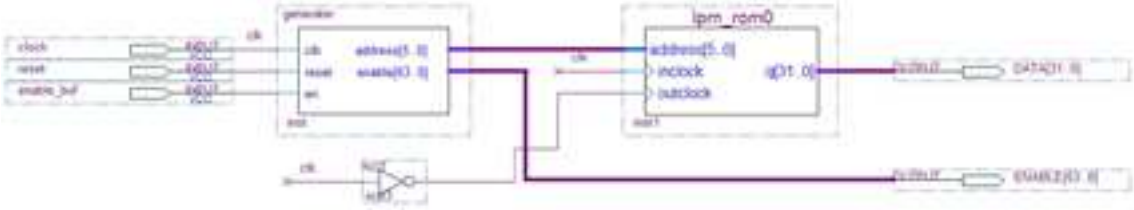
Her bir 8x1N ana bloğu birer adet doğrusal olmayan sigmoid aktivasyon fonksiyonu alt bloğu ve her bir girişin ağırlıklarla çarpılmasını sağlayan sekizer adet çarpıcıya sahip olduğu için FPANN tasarımının en çok kaynak kullanan birimidir. FPANN 32 bit kayan noktalı nümerik formatla işlem gördüğü için çarpıcılar ve toplayıcılar 32x32 boyutludur. Her bir aritmetik işlem bloğunun çıkışı da 32 bit kayan noktalı nümerik formattadır. Çıkış işlemci elemanında yer alan purelin aktivasyon fonksiyonu y = x şeklinde doğrusal bir cevaba sahip olduğundan direk bağlantıyı ifade eder. Bu yüzden herhangi bir donanım kaynağı kullanmaz. Tablo 6.5'te sigmoid, purelin aktivasyon fonksiyonları ve çarpıcı, toplayıcı birimlerinin kaynak kullanımları, toplam lojik eleman, toplam DSP eleman ve toplam hafıza elemanı cinsinden detaylı olarak verilmiştir.

Tablo 6.5. FPANN içerisindeki bazı önemli birimlerin kaynak kullanımları.

	Sigmoid	Çarpıcı	Purelin	Toplayıcı
Toplam Lojik Eleman	1.942	259	0	716
Toplam DSP Eleman	47	7	0	0
Toplam Hafıza Elemanı	892	203	0	283

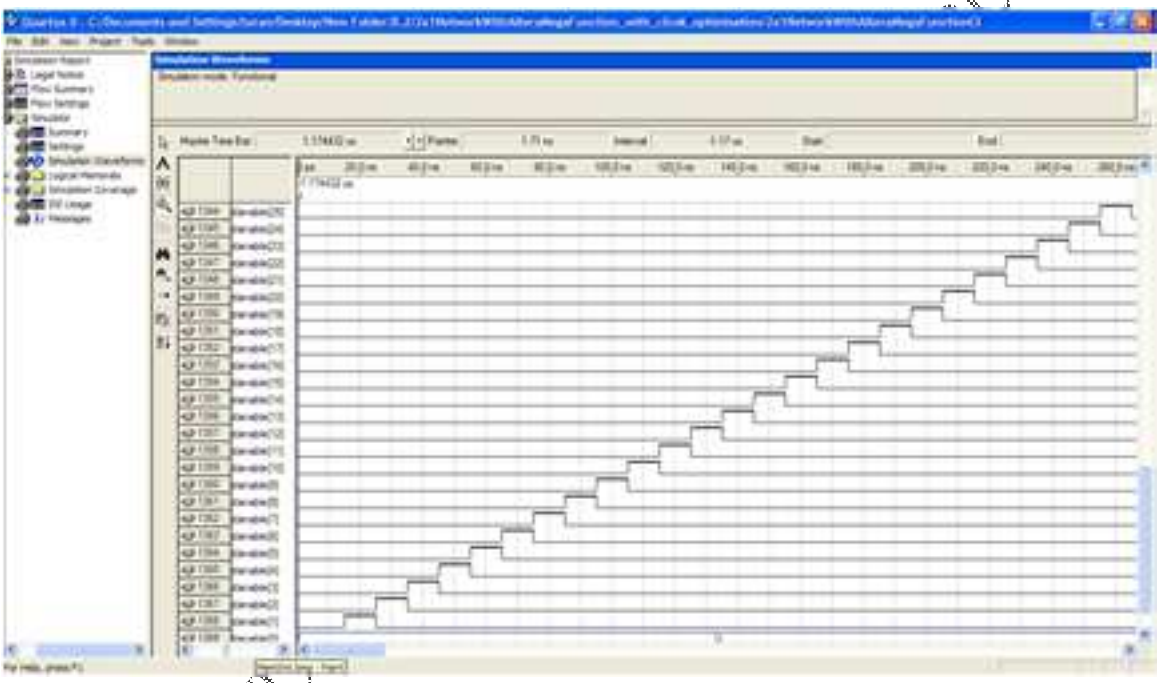
Tasarımın ana bloklarına kısaca göz atılacak olursa, Şekil 6.18'de görüldüğü gibi MemInit bloğu içerisinde 32 bitlik 64 adresli bir hafıza (lpm_rom0 alt bloğu) bulunur ve bu hafıza içerisinde bulunan ağın ağırlık ve giriş değerleri generator alt bloğu tarafından üretilen adres bilgisi ile DATA hattına verilir. DATA hattı bütün ağ içerisine dağılır ve bu hat üzerindeki bilgi 8x1N ve 2x1N blokları içerisindeki tampon alt birimleri tarafından okunur. Hangi tampon'un DATA hattını ne zaman okuyacağı ENABLE çıkışı ile kontrol edilmektedir. Bu çıkış, birim zamanda sadece bir tampon'u aktif etmektedir.

AHMET TURAN ÖZDEMİR - atozdemir@erciyes.edu.tr



Şekil 6.18. MemInit bloğunun içyapısı.

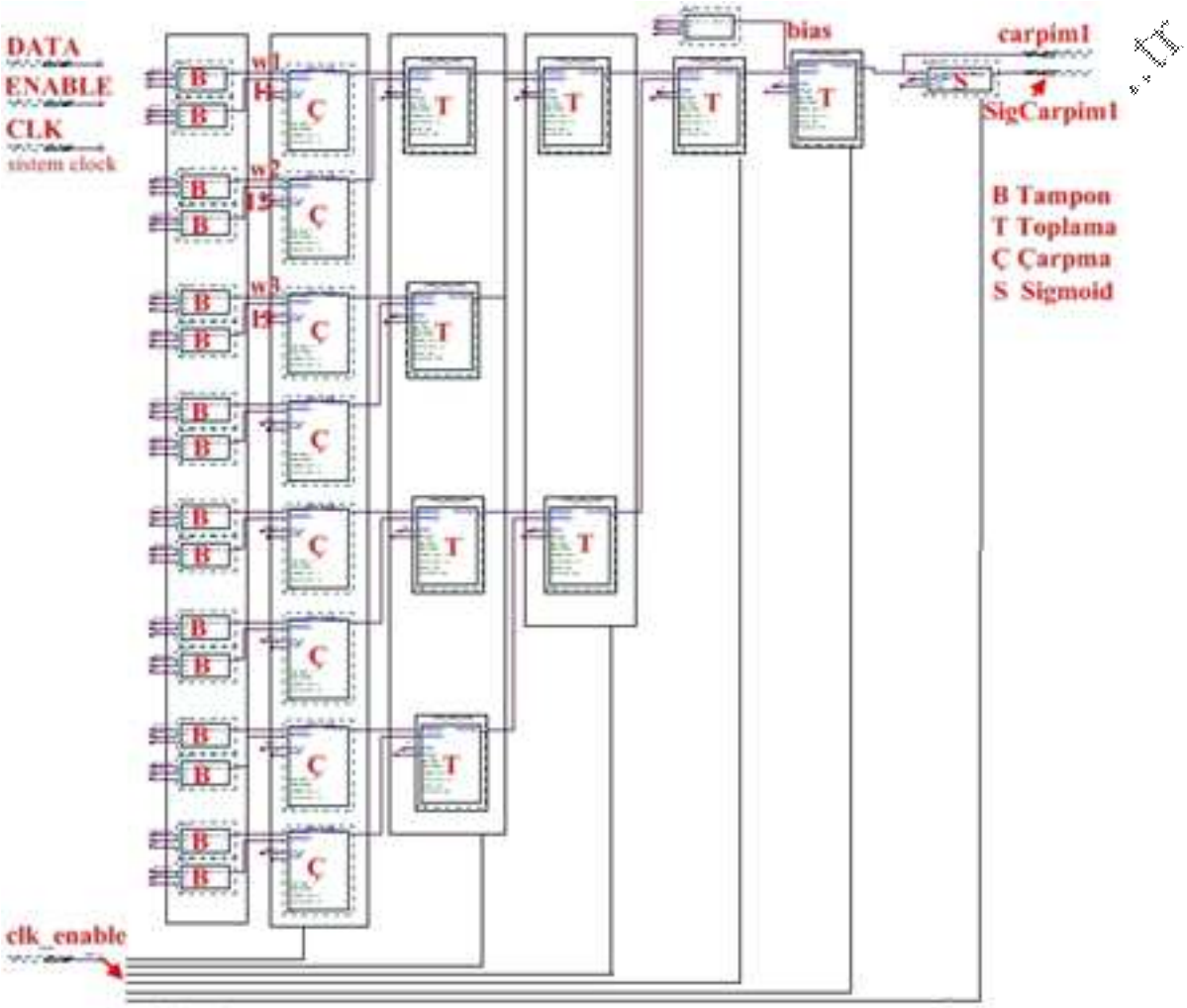
ENABLE verisine ait fonksiyonel benzetim grafiği ise Şekil 6.19'da verilmiştir. Nöronların (8x1N ve 2x1N blokları) içerisindeki tampon'lar ENABLE işareti ile kontrol edilmekte ve ilgili hafıza adresindeki değer, bu sayede ilgili tampon'a yönlendirilmektedir. Bu yöntem ile gömülü hafıza içerisinde saklanan giriş, ağırlık ve bias değerleri okunmaktadır.



Şekil 6.19. Generator bloğu tarafından üretilen ENABLE kontrol işareti.

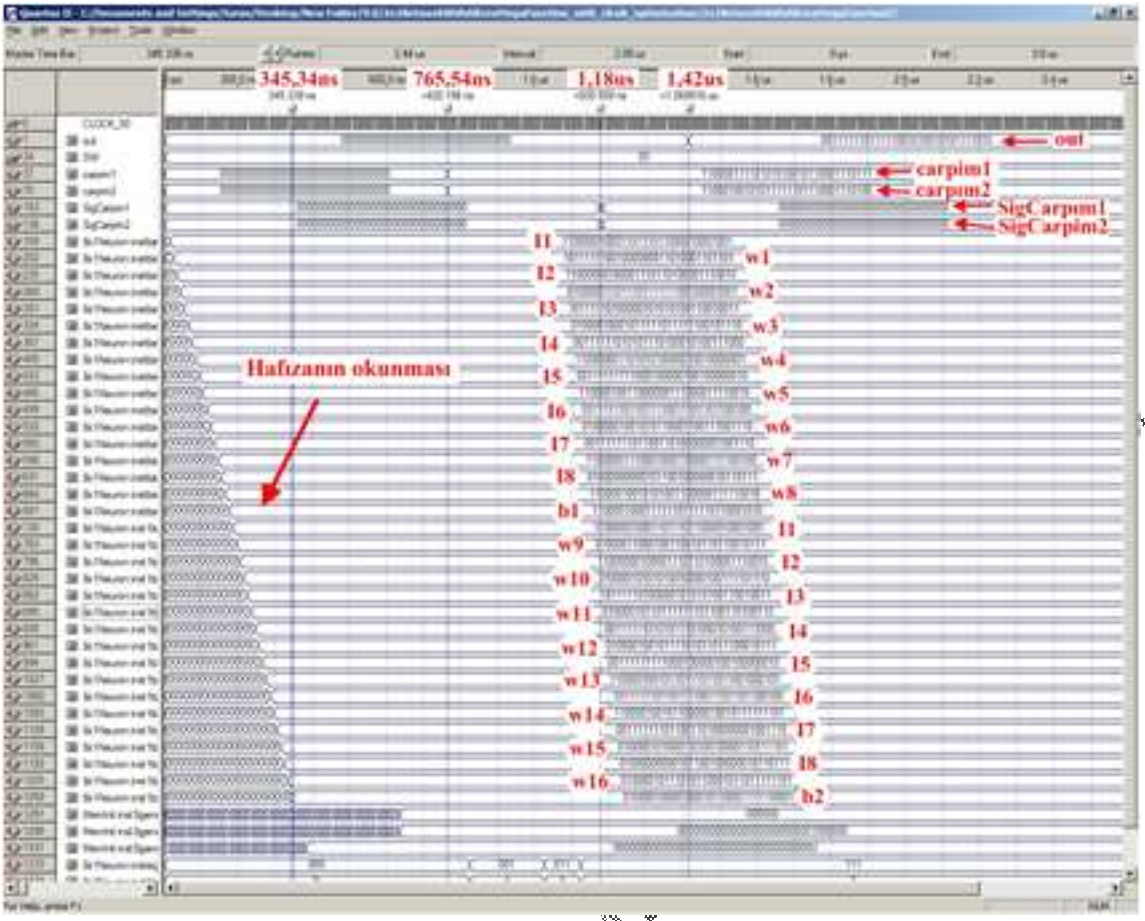
Şekil 6.20'de 8x1N bloğunun içyapısı verilmektedir. Burada tampon'lara gelen DATA verisi ENABLE işareti ile giriş, ağırlık ve bias değerleri şeklinde nörona girişler olarak atanmaktadır. tampon'lardan sonra, giriş değerlerinin çarpıldığı bir çarpıcı blok görülmektedir. Bu çarpıcı bloğun çıkışlarının toplandığı bir toplama katı ve bu toplamların toplandığı ikinci bir toplayıcı katı ve bu toplam sonucunun bias değeri ile toplandığı başka bir toplayıcı katı ve sonucun uygulandığı sigmoid aktivasyon fonksiyonu bloğu görülmektedir. Şekilde görülen clk_enable sinyali işlem birimlerinin devreye alınması ve çıkarılmasını sağlar. Her bir blok tanımlı bir clock periyodu boyunca işlem görmektedir ve bu işlem süresi daha önceden hesaplanarak Şekil 6.16'da verilen clk_man bloğuna VHDL olarak

kodlanmıştır. Bu sayede ilgili blok sistemin clock kontrolünü sağlar. Bu işaret, sistemdeki verinin paralel olarak işlenmesini sağlamaktadır. DATA ve CLK bütün tampon bloklarına uygulanır ancak, Şekil 6.19'da görüldüğü gibi ENABLE sinyali birim zamanda ancak bir tamponun DATA bilgisini hafızaya almasına müsaade eder. Böylece hafızadaki ağ parametreleri ve giriş değerleri doğru bir şekilde çekilerek, ağırlıklar ilgili giriş değerleri ile çarpılmak üzere çarpma birimlerine uygulanır.



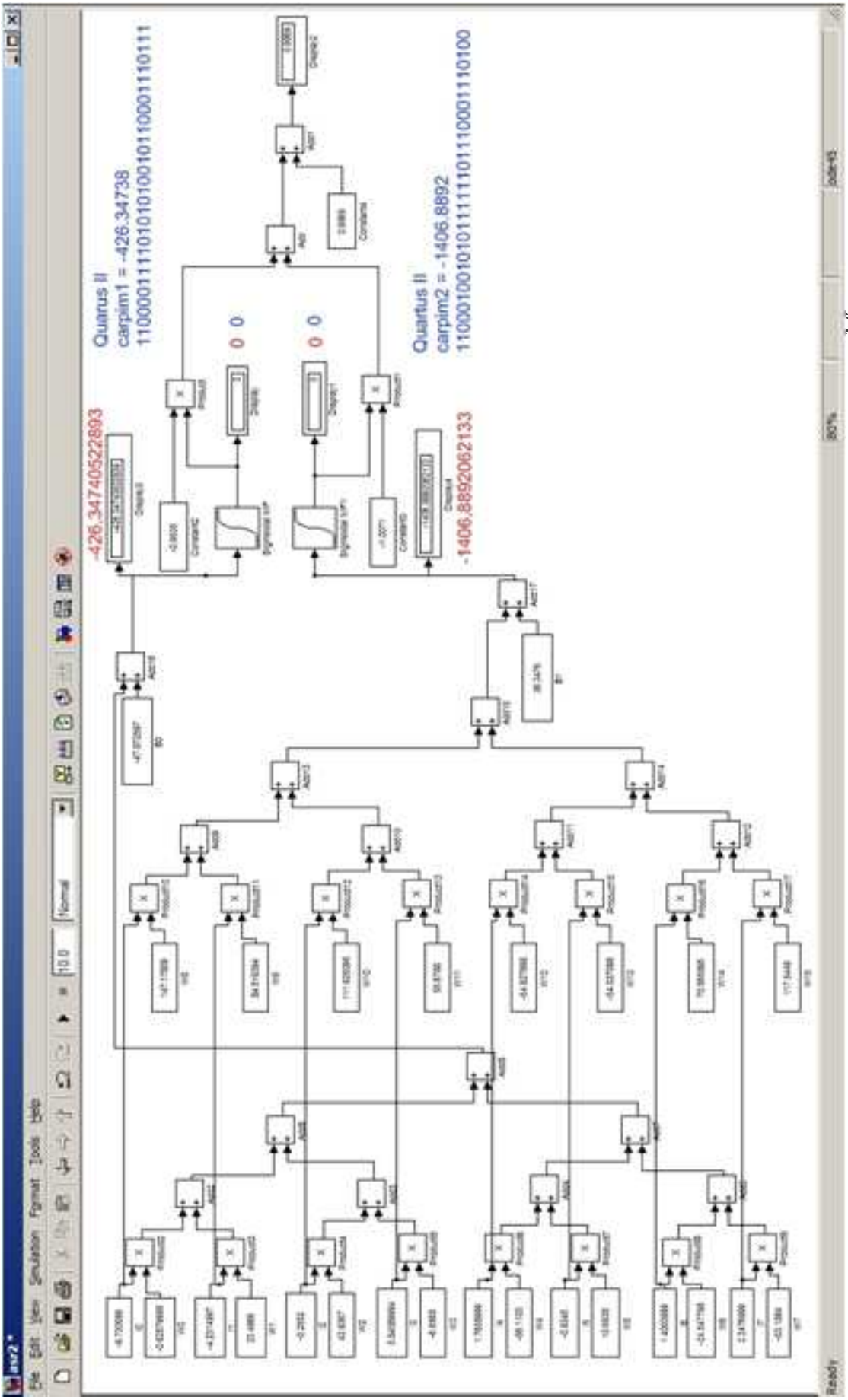
Şekil 6.20. 8x1N bloğunun içyapısı.

Şekil 6.21'de 8x1N bloklarının çıkışlarının aynı anda üretildiği ve bu çıkışların sigmoid aktivasyon fonksiyonu bloğu tarafından yorumlanması için de belli bir süre geçmesi gerektiği görülmektedir. Bütün bu işlem sırası ve süresi clk_man bloğu tarafından üretilen clk_enable tarafından organize edilmektedir.



Şekil 6.21. FPANN donanımının fonksiyonel benzetimi.

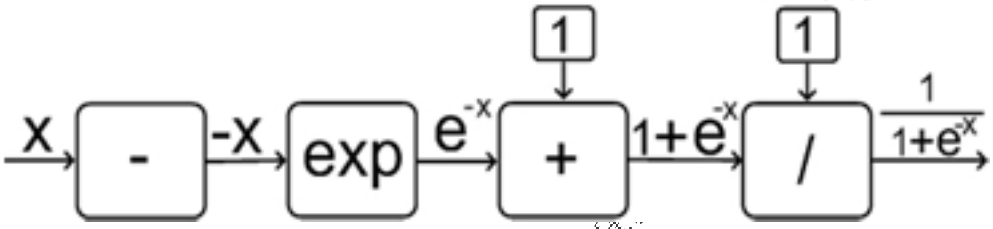
Şekil 6.21'de görüldüğü gibi veriler hafızadan düzenli bir şekilde çekilerek okunmuş ve bu işlem 346 ns sürmüştür. Hafızadan okunan verilerin yani ağırlık ve giriş değerlerinin çarpılarak sonuçlarının toplanması ve bu ağırlıklandırılmış giriş değerlerinin (carpim1 ve carpim2) sigmoid aktivasyon fonksiyonu bloğunun girişine uygulanması işlemi 420 ns sürmüştür. Sigmoid aktivasyon fonksiyonu bloğunun, ağırlıklandırılmış girişleri yorumlaması 410 ns sürmüştür. Gizli katman nöronlarının (8x1N) çıkışları (carpim1 ve carpim2) çıkış nöronuna (2x1N) uygulanmış ve bu katmanın ağın çıkışını üretmesi 240 ns sürmüştür. FPANN ağ donanımı 50 MHz clock frekansı ile 1.42 µs'de hafızadan ağ parametreleri ve 8 giriş verisini okuyarak cevap üretmiştir. FPANN giriş sayısı sabit tutularak gizli katmandaki nöron sayısı arttırılırsa donanımın cevap süresi değişmeyecektir. Görüldüğü gibi carpim1 ve carpim2 değerleri aynı anda elde edilmiştir, çünkü paralel mimari artan nöron sayısından etkilenmemektedir. Bu benzetim sonuçları Şekil 6.22'de verilen aynı ağırlık ve giriş değerlerine sahip bir Simulink modeli ile karşılaştırılmış ve sonuçların aynı olduğu görülmüştür.



Şekil 6.22. FPANN Simulink modeli.

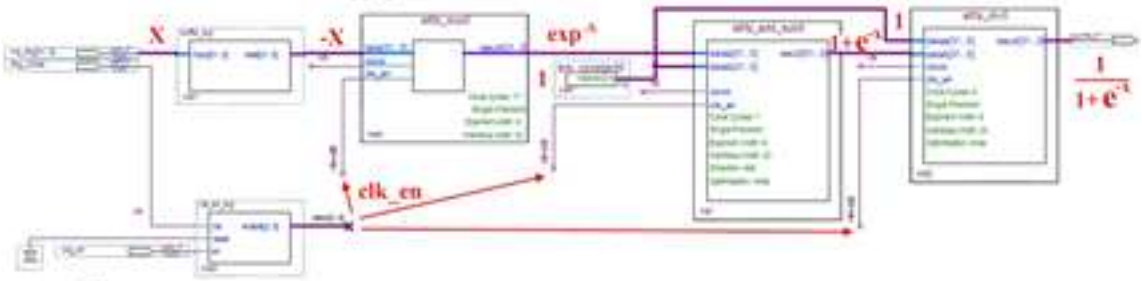
Şekil 6.22'de carpim1 ve carpim2 sinyalleri için Simulink modelinin ürettiği sonuçlar kırmızı, Quartus II benzetim editörünün ürettiği sonuçlar ise mavi ile yazılmıştır. Simulink ve Quartus II sonuçlarının arasında çok küçük bir fark var. Bu fark yuvarlama işleminden kaynaklanmaktadır.

Sigmoid aktivasyon fonksiyonu bloğunun içyapısı Şekil 6.23'te diyagram olarak verilmiştir. Burada 8x1N giriş ve ağırlık değerlerinin çarpımlarının toplamı (örn carpim1) negatife çevrilerek (x giriş değerinden –x değeri elde edilir) exponansiyel işlem bloğuna uygulanır ve y = exp(-x) sonucu üretilir. Bu değer daha sonra sabit 1 ile toplanarak 1+exp(-x) değeri üretilir ve bu değer sabit 1'e bölüm olarak uygulanır. Sigmoid aktivasyon fonksiyonu bloğu üst alma işleminden dolayı çok fazla kaynak kullanmaktadır.



Şekil 6.23. Sigmoid aktivasyon fonksiyonu bloğu blok diyagramı.

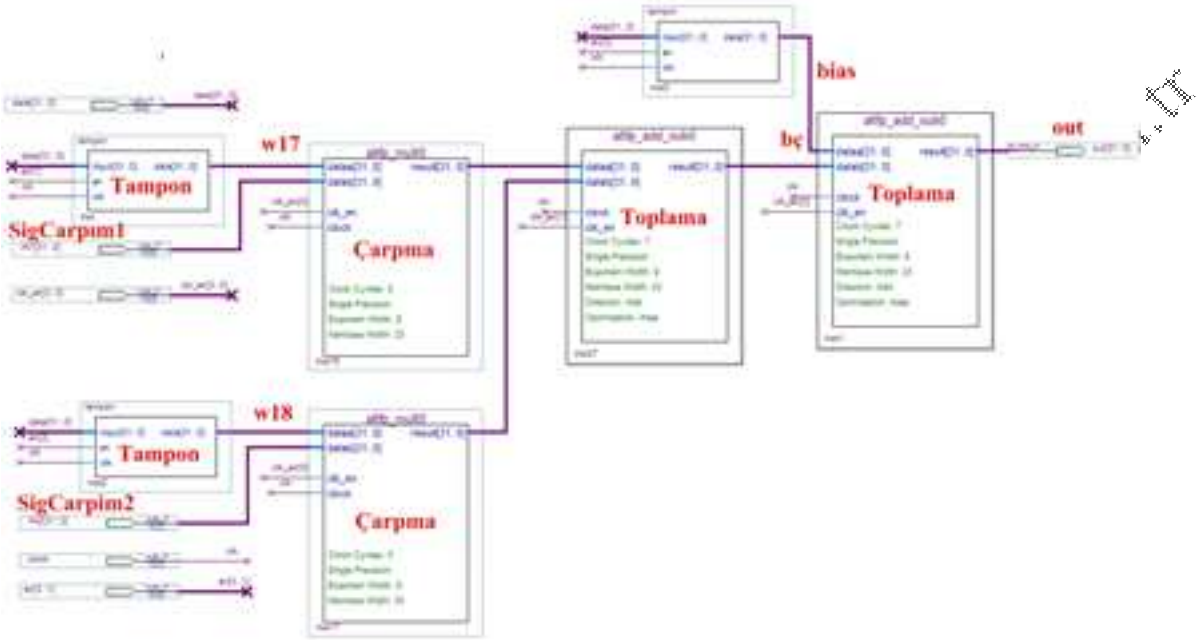
8x1N bloğu içerisinde yer alan sigmoid aktivasyon fonksiyonu bloğunun şematik tasarım dosyası Şekil 6.24'te verilmiştir. Burada X girişinin uygulandığı tampon bloğu, giriş ifadesinin işaret bitini değiştirerek 32 bit kayan noktalı nümerik ifadenin – (eksi) ile çarpımını sağlar. Üst alma operatörü Altera Mega Function kütüphanesinden alınarak kullanılmıştır.



Şekil 6.24. Sigmoid aktivasyon fonksiyonu bloğunun içyapısı.

Daha önce de bahsedildiği gibi sistem içerisindeki 8x1N bloklarının çıkışlarının uygulandığı 2x1N bloğunda aktivasyon fonksiyonu olarak purelin y = x fonksiyonu kullanılmıştır. Bu fonksiyon girişi hiçbir değişiklik olmadan çıkışa aktarmaktadır yani sadece bir bağlantıdır.

Şekil 6.25'te 2x1N bloğunun içyapısı şematik olarak verilmiştir. Burada tamponlarda tutulan ağırlık değerleri 8x1N bloğu tarafından üretilen çıkış değerleri ile çarpılır, bu verilerin toplamı bir bias değeri ile toplanarak çıkışa aktarılır. 2x1N bloğu içerisindeki alt birimlerin çalışması yine clk_enable sinyali tarafından kontrol edilmektedir.



Şekil 6.25. 2x1N bloğunun içyapısı.

32 bit kayan noktalı nümerik tanımlama formatı aritmetik işlemlerde çok yüksek derecede doğrulukla cevap üretirken, donanım gerçeklemelerinde kaynak kullanımını önemli ölçüde arttırmaktadır. Kaynak kullanımını arttırmasına rağmen yüksek hassasiyetin gerektiği uygulamalarda kayan noktalı nümerik tanımlama formatı sıklıkla tercih edilmektedir. FPANN EP3C120F780 yongasının toplam lojik elemanlarının %19'unu ve gömülü çarpıcılarının ise %38'ini kullanmıştır. Bir sonraki başlıkta FXANN tasarımı incelenecek ve daha sonra bu iki yapı kaynak kullanımları bakımından incelenecektir.

6.4.2. 16 Bit Sabit Noktalı Nümerik Tanımlama ile FPGA Üzerinde YSA Donanımı Gerçekleştirilmesi (FXANN)

FXANN işaret bitli 16 bit sabit noktalı nümerik tanımlamayı kullanmaktadır. 16 bitin ilk biti işaret biti, takip eden yedi biti tamsayı değeri ve kalan sekiz biti ise virgülden sonraki hassasiyeti tanımlamak için kullanılmıştır. 16 bit nümerik tanımlama ±128 nümerik aralığını 2^{-8} hassasiyet ile tanımlamaktadır. Bu nümerik tanımlama normalize edilmiş veri setinin

hesaplamasını sağlayacak yeterlilikte bir nümerik aralığa sahiptir. FXANN donanımının Matlab modelinde elde edilen ağ parametreleri Tablo 6.6'da verilmiştir. 5 adet kalp atımı için TBA işlemi sonucunda elde edilen ilk sekiz temel bileşenin (TB) nümerik değerleri ise Tablo 6.7'de verilmiştir. Temel bileşenler ve ağ parametrelerinin çarpım veya toplamlarından hesaplanacak nümerik değerlerin 16 bit sahasında ifade edileceği ve nümerik değerlerde taşmalar oluşmayacağı açıkça görülmektedir.

Tablo 6.6. 8x2x1 boyutlu YSA için ağ parametreleri FPANN.

w1	=	-1,6323	*w9*	=	-3,8596	*w16*		-2,1545
w2	=	-3,0263	*w10*	=	-1,5485	*w17*		1,7458
w3	=	-1,3357	*w11*	=	1,2032	*b1*	=	0,8821
w4	=	1,1289	*w12*	=	-1,5076	*b2*	=	-8,4431
w5	=	3,8375	*w13*	=	0,9262	*bç*	=	0,9886
w6	=	-3,2129	*w14*	=	2,5759			
w7	=	0,9284	*w15*	=	-0,3038			
w8	=	0,5648	*w16*	=	-2,0272			

Tablo 6.7. İlk sekiz temel bileşen.

TB'ler	1. Kalp Atımı	2. Kalp Atımı	3. Kalp Atımı	4. Kalp Atımı	5. Kalp Atımı
TB1	-0,3249	-0,9469	-0,2755	-0,4913	-1,1727
TB2	-0,3337	0,5849	-0,5448	-0,0604	1,0796
TB3	-0,0275	-0,0428	-0,0092	-0,0789	-0,0963
TB4	-0,0731	-0,1485	0,0375	-0,0978	-0,1431
TB5	-0,0794	-0,01397	-0,0620	-0,0138	0,0574
TB6	0,1506	0,0821	0,1778	0,1621	0,0862
TB7	-0,0815	-0,0582	-0,0122	-0,1038	-0,2057
TB8	-0,3379	-0,3044	-0,3471	-0,3450	-0,2335

FXANN donanımı aktivasyon fonksiyonu olarak Parçalı Doğrusal Sigmoid Aktivasyon (PWLAN) fonksiyonunu kullanmaktadır. 16 bit sabit noktalı nümerik tanımlama doğrusal olmayan aritmetik işlemlerin hesaplanmasında direk olarak kullanılamamaktadır. Bu yüzden sigmoid aktivasyon fonksiyonu bir eğri uydurma yöntemi olan PWLAN ile yedi parçalı doğrusal olarak Denklem 6.2'de verildiği gibi formüle edilmiştir. Burada *f*(X) PWLAN aktivasyon fonksiyonunun çıkış cevabıdır ve 0 ile 1 arasında değerler alır. FPANN için

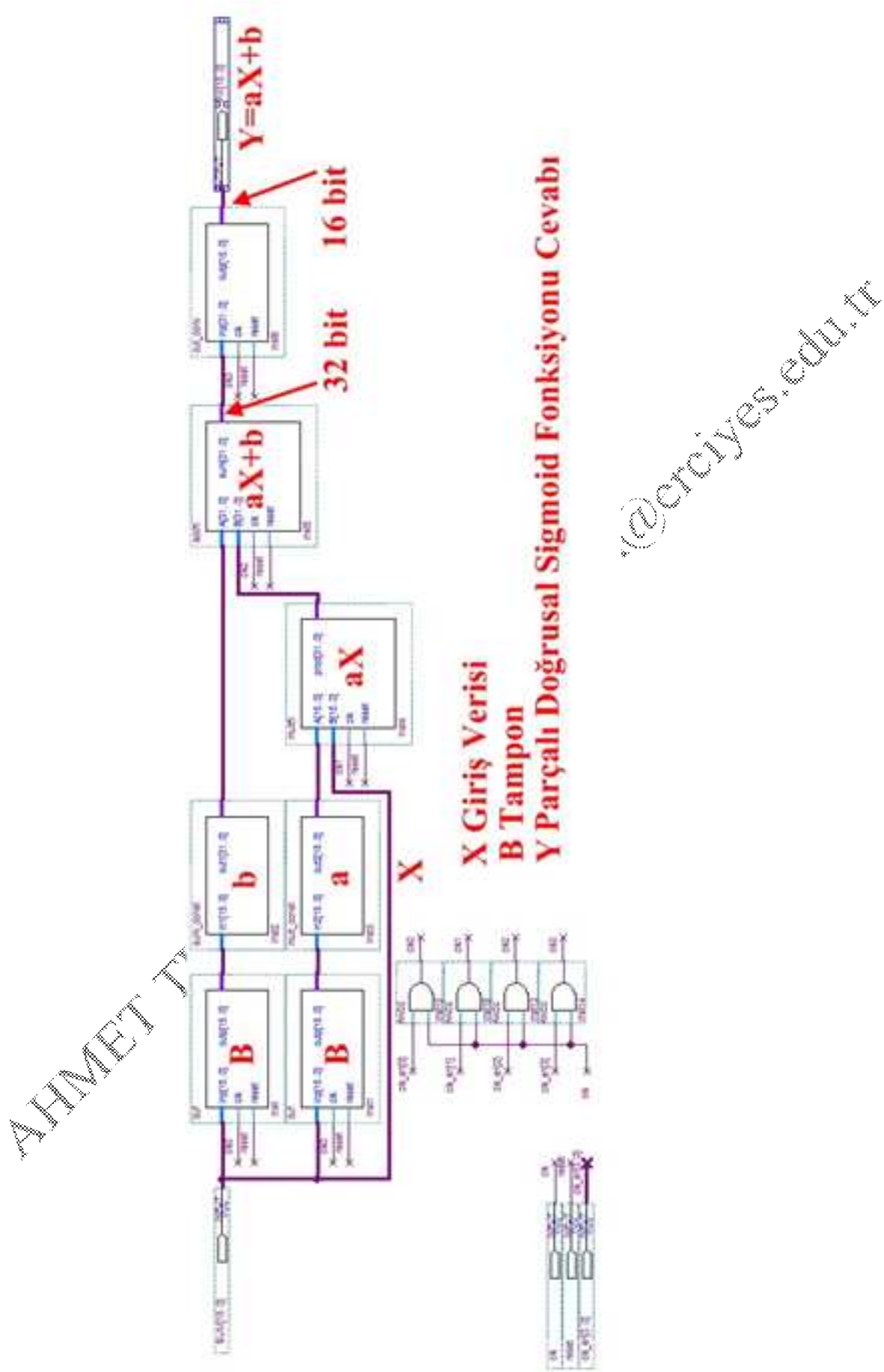
AHMET TURAN ÖZDEMİR, aturan@erciyes.edu.tr

PWLAN kuralı, VHDL dil editörü kullanılarak, 16 bit sabit noktalı nümerik format ile FPGA üzerinde oluşturulmuştur.

$$
\begin{array}{lrcccl}
f(X) = 1 & 5 & \leq & X & & \\
f(X) = 0{,}03125X + 0{,}84375 & 2{,}375 & \leq & X & < & 5 \\
f(X) = 0{,}125X + 0{,}625 & 1 & \leq & X & < & 2{,}375 \\
f(X) = 0{,}25X + 0{,}5 & -1 & < & X & < & 1 \\
f(X) = 0{,}125X + 0{,}375 & -2{,}375 & < & X & \leq & -1 \\
f(X) = 0{,}03125X + 0{,}15625 & -5 & < & X & \leq & -2{,}375 \\
f(X) = 0 & & & X & \leq & -5
\end{array}
\qquad (6.2)
$$

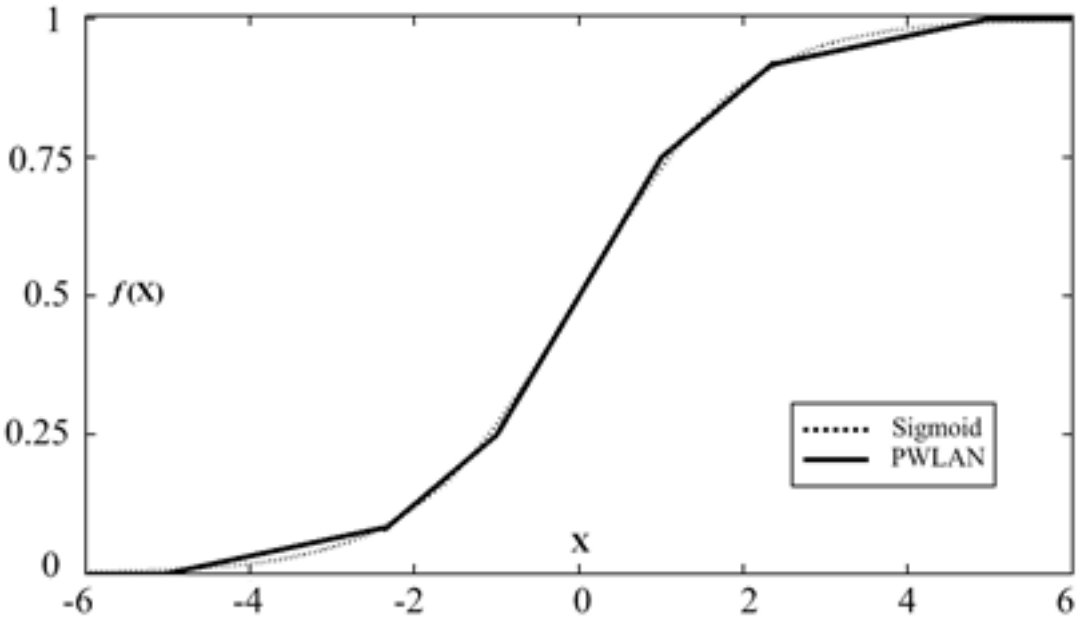
8x2x1 boyutlu YSA modeli, FXANN gerçeklemesinden önce yukarıda denklemi verilen PWLAN aktivasyon fonksiyonu kullanılarak Matlab ortamında eğitilmiştir. Çünkü parçalı doğrusal yaklaşım tam olarak sigmoid fonksiyonunun cevabını vermediği için yapay sinir ağının eğitimin PWLAN ile yapılması gereklidir. Aksi takdirde eğitim hatasına ilave olarak bu iki fonksiyonun fark hatası da YSA'nın sınıflandırma hatasına eklenecektir. PWLAN fonksiyonu Matlab ortamında bir komut dosyası olarak oluşturulduktan sonra eğitim esnasında çağrılarak 8x2x1 boyutlu referans YSA'nın eğitiminde kullanılmıştır. $f(X) = aX+b$ şeklinde yedi parça olarak ifade edilen PWLAN fonksiyonu Quartus II'de Şekil 6.26'daki gibi oluşturulmuştur.

AHMET TURAN ÖZDEMİR atuzdemir@yes.edu.tr



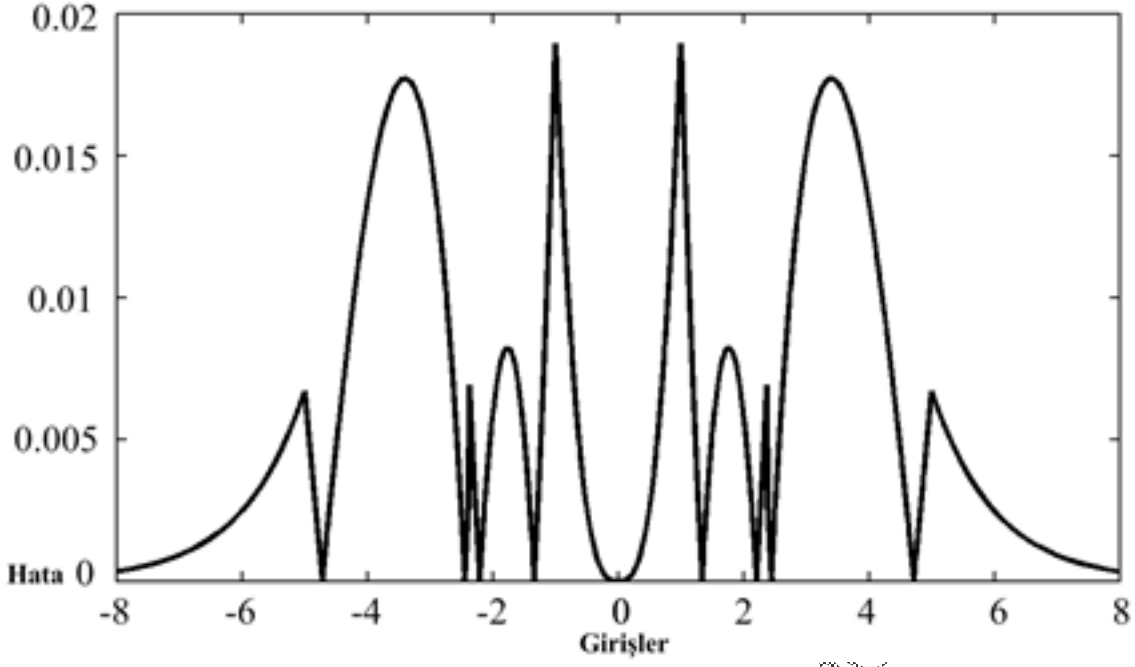
Şekil 6.26.WLAN bloğunun içyapısı.

PWLAN yedi parçalı yaklaşım ile [-8 8] aralığı için X giriş verisine bağlı olarak PWLAN çıkış cevabı maksimum %1,89 hata ile gerçek sigmoid fonksiyonunun çıkış cevabına yakınsamaktadır. Bu değer %2'nin altında kabul edilebilir bir hata oranıdır. Gerçek sigmoid fonksiyonu ile parçalı doğrusal yaklaşım arasındaki fark hatası, bu çalışmada yedi adet kullanılan aralıkların sayısının arttırılması ile daha da azaltılabilir. PWLAN fonksiyonunun [-8 8] aralığında çıkış karakteristiği ve gerçek sigmoid fonksiyonu ile arasındaki fark hatasının grafiksel gösterimleri sırası ile Şekil 6.27 ve Şekil 6.28'de verilmiştir.



Şekil 6.27: PWLAN fonksiyonunun gerçek sigmoide yakınsaması.

FPANN donanımında kullanılan gerçek sigmoid ile FXANN donanımında kullanılan PWLAN aktivasyon fonksiyonlarının birbirlerinden çıkarılması ile elde edilen fark Şekil 6.28'de görüldüğü gibi %2'den azdır. X giriş verisi ekseni [-8 8] aralığında verilmiştir. Çünkü bu aralığın dışındaki giriş değerleri logaritmik sigmoid fonksiyonunu sırası ile 0 ve 1 değerlerine satüre etmektedir. Bu yüzden tanımlı aralığın dışındaki değerler için hata oranı %0 civarlarındadır. Eğitimler esnasında kullanılan parçalı doğrusal sigmoid aktivasyon fonksiyonunun Matlab kodları Ekler kısmında verilmiştir.



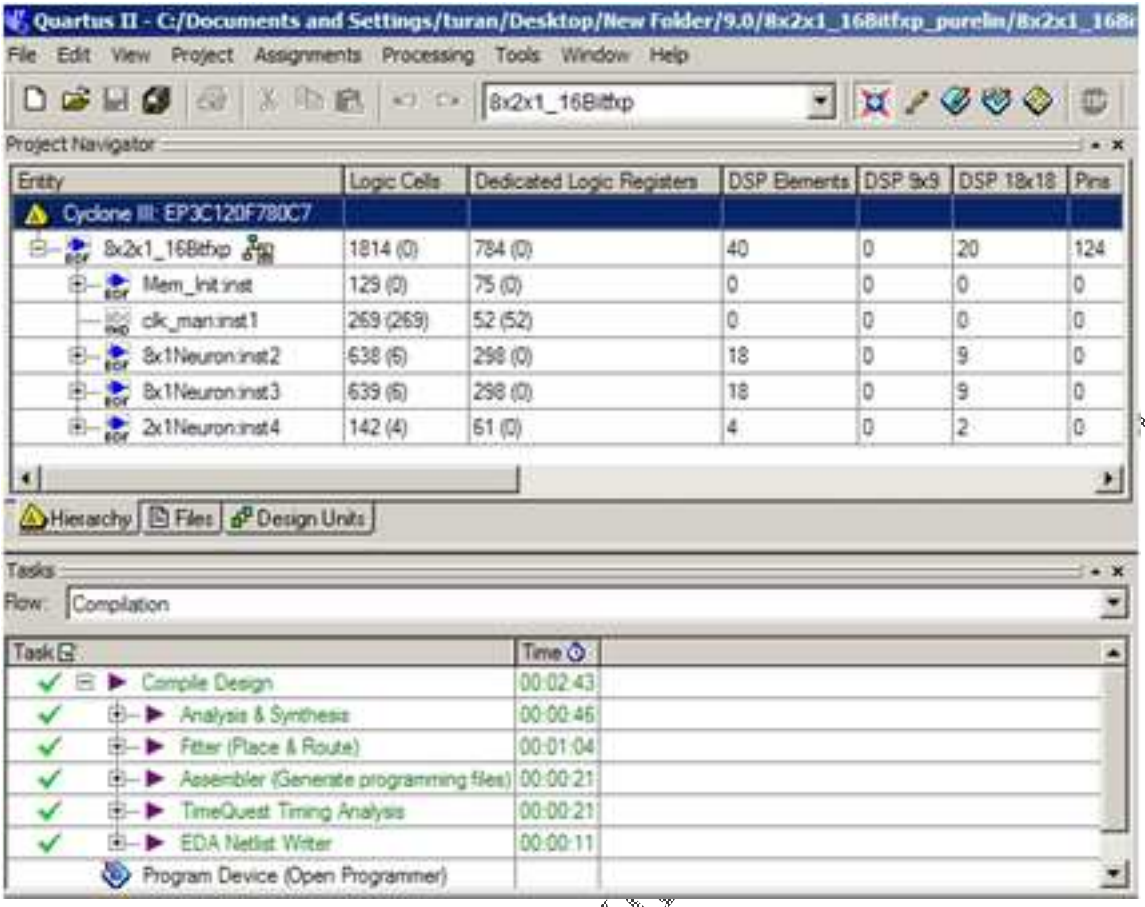
Şekil 6.28. PWLAN fonksiyonunu ile gerçek sigmoid arasındaki fark.

PWLAN ile gerçek sigmoid aktivasyon fonksiyonu arasındaki karakteristik fark oldukça küçük olmasına karşın, PWLAN aktivasyon fonksiyonu oldukça az yonga alanı kullanmaktadır. Kaynak kullanımının azalmasında 16 bit nümerik tanımlamanın çarpma ve diğer aritmetik işlemleri basitleştirmesinin etkisinin yanında, PWLAN fonksiyonunda bölme ve üst alma işlemlerinin olmamasının da bu azalmada büyük bir katkısı vardır. Tablo 6.8'de FXANN donanımında kullanılan PWLAN ve purelin aktivasyon fonksiyonları ile çarpıcı ve toplayıcı elemanların kaynak kullanımları verilmiştir. Tablodan görüleceği üzere PWLAN aktivasyon fonksiyonu birimi sadece iki adet DSP elemanı kullanmaktadır. Bu iki adet DSP elemanı ise PWLAN birimi içerisinde yer alan çarpma birimi tarafından kullanılmaktadır.

Tablo 6.8. FXANN içerisindeki bazı önemli birimlerin kaynak kullanımları.

	PWLAN	**Çarpıcı**	**Purelin**	**Toplayıcı**
Toplam Lojik Eleman	73	0	16	32
Toplam DSP Eleman	2	2	0	0
Toplam Hafıza Elemanı	42	0	16	32

FXANN donanımının kaynak kullanımı ve onaylanmış tasarım akışının bulunduğu ekran görüntüsü Şekil 6.29'da verilmiştir. FPANN ve FXANN aynı mimarinin iki farklı nümerik tanımlama ile oluşturulmuş halleri olduğundan ağı oluşturan bloklar aynıdır.



Şekil 6.29. Kaynak kullanımları ve tasarım süreçlerinin onayı (FXANN).

FXANN 4'ü gizli katman işlemci elemanlarında bulunan aktivasyon fonksiyon blokları ve 36'sı sistemde yer alan 18 çarpım elemanı tarafından kullanılan toplam 40 DSP işlem elemanına ihtiyaç duymaktadır. Sonuç olarak 40 DSP işlemci elemanının 36 tanesi 8x1N bloklarında 4 tanesi ise 2x1N bloğunda kullanılır. FXANN tasarlanırken gizli katmanlarda kullanılan aktivasyon fonksiyonları, yedi parçalı doğrusal PWLAN sigmoid fonksiyonu olarak seçildiği daha önce de ifade edilmişti. Her bir PWLAN iki adet DSP işlemci elemanı kullanır. Bu iki DSP işlem elemanı, PWLAN bloğu içerisinde bulunun bir çarpıcı tarafından kullanılmaktadır. 8x2x1 boyutlu YSA modelinin 16 bit nümerik tanımlama kullanılarak FPGA üzerinde tasarlanması ile kaynak kullanımı çok önemli ölçüde azaltılmıştır. Tablo 6.9'da, FXANN ve içerisindeki dört ana bloğun donanım kaynak kullanımları, toplam lojik eleman, toplam DSP işlemci elemanı ve hafıza kaynakları cinsinden, adet ve toplam kullanımın yüzdeleri cinsinden ayrıntılı olarak verilmiştir.

Tablo 6.9. FXANN ve ana bloklarının donanım kaynak kullanımları.

	2 8x1N	2x1N	MemInit	ClockMan	FXANN
Toplam Lojik Eleman	1.277 70%	142 8%	129 7%	269 15%	1.814 100%
Toplam DSP Eleman	36 90%	4 10%	0	0	40 100%
Toplam Hafıza Elemanı	596 75%	61 8%	75 10%	52 7%	784 100%

**FXANN iki adet 8x1 bloğuna sahiptir*

Tablo 6.5 ve Tablo 6.8 karşılaştırıldığında FPANN daha fazla donanım kaynagi tüketiyor olmasına karşın, purelin aktivasyon fonksiyonu için hiç lojik, DSP veya hafıza elemanı kullanmadığı halde FXANN 16'şar adet lojik ve hafıza elemanı kullanmıştır. Bunun sebebi y=x karakteristiğindeki purelin fonksiyonu için herhangi bir donanıma ihtiyaç duyulmamasıdır. FPANN donanımında 32 bit nümerik değer bu bloğa geldiği haliyle çıkışına transfer edilir. Yani bu blok ileri yönlü bir bağlantıdır. FXANN donanımında kullanılan lojik ve hafıza elemanlarının sebebi ise çıkış katmanındaki işlemci elemanda yani 2x1N bloğunda, 16 bitlik ağırlık değerleri ile gizli katmandaki 8x1N bloklarından gelen 16 bitlik giriş değerlerinin çarpılması sonucunda oluşan 32 bitlik nümerik ifadenin 16 bit ile ifade edilmesi için kullanılan bir alt birimdir. Bu çevrim işlemi sonucunda eğer taşma oluşursa bir hata kodu üretilerek işlem sonlandırılır ve bir diğer atım için hesaplama yapılır.

16 bit ile tanımlamayacak kadar büyük olan değerler ±128 değerine çevrilerek kullanıldığında da sonuçlar genellikle değişmemekte ve doğru çıkmaktadır. Çünkü sigmoid aktivasyon fonksiyonu [-6 ile 6] arasında duyarlıdır ve [-8 ile 8] aralığının dışına çıkıldığında, sigmoid fonksiyonunun çıkışı [0 veya 1] değerine satüre olmaktadır. ±128 değerleri satürasyon bölgelerindedir ve bu değerlerin ürettikleri sonuçlar genellikle doğru çıkmaktadır.

6.5. FPANN ve FXANN Donanımlarının Kaynak Kullanımları Bakımından Karşılaştırılması

16 bit nümerik tanımlama ve parçalı doğrusal sigmoid aktivasyon fonksiyonunun kullanılması ile kaynak kullanımı başka bir ifade ile donanım maliyeti çok büyük ölçüde azaltılmıştır. Tablo 6.10'da FPANN ve FXANN donanımları ve bu donanımlar içerisinde yer alan temel bloklarının kaynak kullanımları karşılaştırmalı olarak verilmiştir. Bu tablonun en son sütununda yer alan boyut sekmesi, 16 bit nümerik tanımlama ve PWLAN aktivasyon

AHMET BURAK ÖZDEMİR - ozdemirbilgisayar.tr

fonksiyonunun kullanılması ile FXANN donanımının ne ölçüde küçültüldüğü hakkında çarpıcı sonuçlar vermektedir. FXANN donanımı toplam lojik eleman sayısı bakımından 12,78, toplam DSP eleman sayısı bakımından 5,5 ve toplam hafıza elemanı sayısı bakımından 13,8 defa FPANN donanımından daha küçüktür.

Tablo 6.10. Temel blokların kaynak kullanımları bakımından karşılaştırılması.

		8x1N*	2x1N	MemInit	ClockMan	Toplam	Boyut
Toplam Lojik Eleman	FPANN	20.690	2.118	125	256	23.189	1 br*
	FXANN	1.276	142	129	268	1.814	0,078 br
Toplam DSP Eleman	FPANN	206	14	0	0	220	1 br
	FXANN	36	4	0	0	40	0,18 br
Toplam Hafıza Elemanı	FPANN	9.646	1.054	75	41	10.816	1 br
	FXANN	596	61	75	52	784	0,072 br

**8x1N bloğu iki adet kullanılmıştır ve tabloda verilen değer toplamı ifade eder.*
**br "birim" kelimesinin kısaltması olarak kullanılmıştır. FPANN tasarımında 1 birim alanda gerçekleştirilen donanımların FXANN tasarımında kaç birimlik alana sığdırıldığı vurgulanmak istenmiştir.*

Nümerik tanımlamalarda 32 bit yerine 16 bit kullanılması, işlem gören veri boyutunu azalttığı için FXANN donanımında kullanılan toplam lojik ve hafıza elemanlarında çok ciddi ölçüde azalma meydana gelmiştir. Veri boyutundaki azalma DSP eleman sayısına da etki etmekle beraber, DSP eleman sayısındaki azalmanın en önemli sebebi sabit noktalı nümerik hesaplama algoritmasının, kayan noktalı nümerik hesaplama algoritmasına göre daha basit olmasıdır. Ayrıca PWLAN fonksiyonunun karmaşık sigmoid fonksiyonuna göre daha basit bir aritmetik formülizasyona sahip olması da donanım kaynak kullanımlarının azalması yönünde oldukça etkili olmuştur. Bütün bu etkiler FXANN donanımının daha basit olması ve az sayıda yonga alanı kullanması yönünde etkili olmuşlardır.

PWLAN fonksiyonu ve 16 bit nümerik sayı formatının aritmetik birimler üzerinde kaynak kullanımına olan etkisinin daha iyi anlaşılabilmesi için aritmetik birimlerin kaynak kullanımları Tablo 6.11'de detaylı bir şekilde verilmiştir. Bu tabloda kaynak kullanımına ciddi etkileri olan aktivasyon fonksiyonları (Sigmoid ve PWLAN), çarpma bloğu, sayı formatlarından çok fazla etkilenmeyen purelin aktivasyon fonksiyonu bloğu ve toplama alt bloğu kaynak kullanımları bakımından incelenmiştir. Bu sayede nümerik sayı formatlarının hangi aritmetik işlemlere daha duyarlı olduğu da görülebilir.

AHMET TURAN GÜLDEMİR, atuzgeriyes.edu.tr

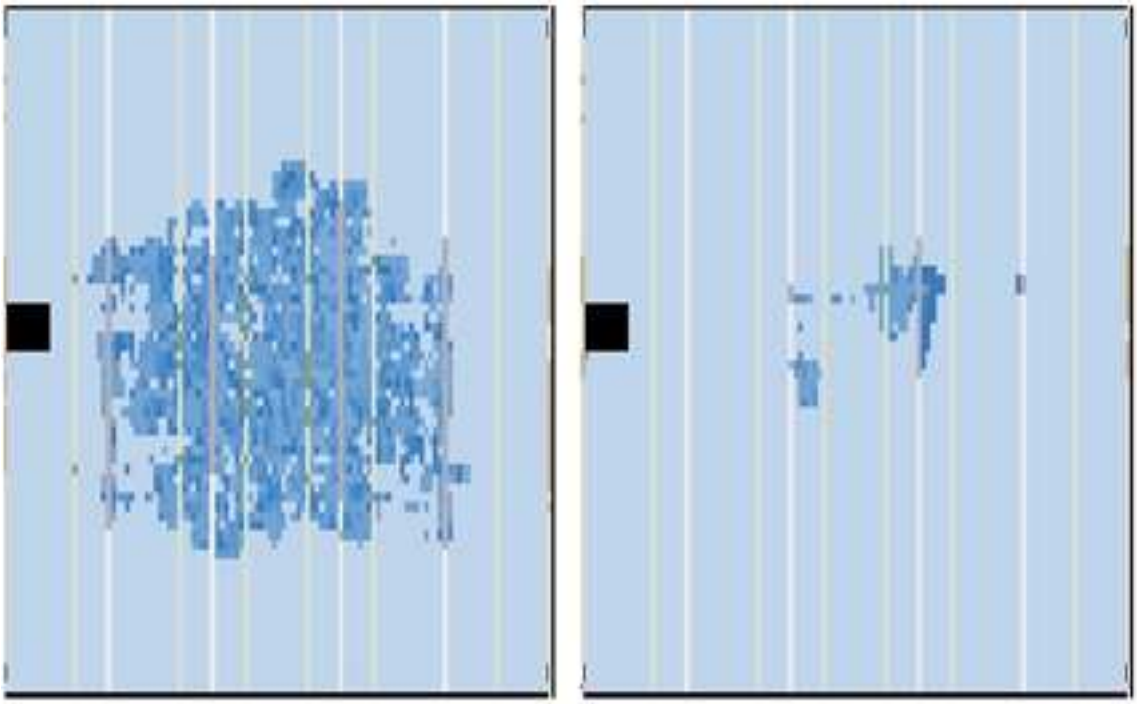
Tablo 6.11. Aritmetik birimlerin kaynak kullanımları bakımından karşılaştırılması.

			AF*	Çarpıcı	Purelin	Toplayıcı
Toplam Lojik	FPANN	*Sigmoid*	1.942	259	0	716
Eleman	FXANN	*PWLAN*	73	0	16	32
Toplam DSP	FPANN	*Sigmoid*	47	7	0	0
Eleman	FXANN	*PWLAN*	2	2	0	0
Toplam Hafıza	FPANN	*Sigmoid*	892	203	0	283
Elemanı	FXANN	*PWLAN*	42	0	16	32

**AF gizli katmandaki işlemci elemanlarda kullanılan aktivasyon fonksiyonudur. AF FPANN için sigmoid FXANN için PWLAN aktivasyon fonksiyonudur.*

Tasarımcı bir sistemi tasarlarken, tasarımın ihtiyaç duyduğu donanım kaynağı yoğunluklarını tespit ederek buna uygun bir yonga seçer. Yonganın sahip olduğu donanım yoğunlu maliyete etki eden en önemli parametredir. FPGA üzerinde oluşturulacak olan bir YSA'nın boyutu, kullanılacak olan FPGA yongasının yoğunluğuna bağlıdır ve bu sınırlama tasarımcıyı daha küçük boyutlu ağlar oluşturmaya zorlamaktadır. Bununla birlikte IEEE 754 2008 32 bit kayan noktalı nümerik tanımlama yerine, 16 bit sabit noktalı nümerik tanımlama yöntemi kullanarak fazla miktarda kaynak kullanımına sahip aktivasyon fonksiyonu ve çarpıcı gibi elemanların boyutları küçültülerek, aynı yonga içerisinde daha büyük ölçekli YSA donanımları oluşturmak veya ilgili YSA'yı daha küçük/ucuz bir yonga üzerinde oluşturmak mümkündür. Altera Quartus II FPGA editörünün yonga içerisinde kullanılan donanım kaynaklarını gösteren Chip Planner eklentisinin FPANN ve FXANN donanımlarının FPGA yongasının ne kadarını kullandığı sırası ile Şekil 6.30 (a) ve (b)'de verilmiştir. FPANN ve FXANN donanımları Cyclone III EP3C120F780 yongası üzerinde gerçeklenmiştir [147].

AHMET TURAN ÖZDEMİR - oturan@erciyes.edu.tr



Şekil 6.30. a) FPANN Chip Planner çıktısı, b) FXANN Chip Planner çıktısı.

FXANN tasarımı bloklar halinde verilmemiştir, çünkü bloklar FPANN tasarımında kullanılan bloklar ile aynı işlevselliğe sahiptir. Şematik gösterimlerindeki tek farklılık veri yollarının boyutudur. FPANN'de 32 bit veri yolları kullanılırken, FXANN'de 16 bit genişlikli veri yolları kullanılmıştır. Hafıza elemanları ve çarpım terimleri de aynı şekilde 32 bitten 16 bite düşmüştür. Ancak FXANN tasarımında, 16 bit sabit noktalı nümerik tanımlama ile gerçekleştirilen çarpım işlemlerinin sonuçları 32 bit çıktığı için bu sonuçlar 16 bite yuvarlanmıştır.

AHMET TURAL ÖZDEMİR atu.tr