ÇİP TASARIM YARIŞMASI
SAYISAL İŞLEMCİ TASARIM KATEGORİSİ
ÖN TASARIM RAPORU ŞABLONU

TAKIM ADI: YILDIRIM MİKROTEK
BAŞVURU ID: 292814

2024

1. Giriş

Bu proje kapsamında RV32IMAFB_Zicsr buyruk kümesi mimarisine sahip RISC-V tabanlı bir işlemci tasarlanacak olup işlemci özelindeki tasarım kararları bu raporda açıklanmaktadır. Tasarlanacak işlemci UART çevre birimi ve 4 KB'lık birinci seviye önbelleğe sahip olmasıyla birlikte 4 aşamalı boru hattı ile işlemesi planlanmaktadır. Proje boyunca ön tasarım ve tasarım aşamalarında verilecek kararlar test edilirken Vivado [1], Verilator [2] ve coco.tb [3] gibi test ve tasarım araçlarından yardım alınacaktır. Sentez ve simülasyonlarda kullanmak üzere Synopsys Design Compiler [4] aracının kullanımına karar verilmiştir. Sistemin tasarımı tamamlandığında, işlemci Zybo Z7 XC7Z010 FPGA üzerinde test edilecek, Teknofest tarafından belirlenen EDA aracı kullanılarak sistemin ASIC fiziksel tasarımı oluşturulacaktır. Bu süreçte tüm takım üyelerinin hem FPGA üzerinde modelleme hem de ASIC tasarımı konularında tecrübe edinmesi hedeflenmektedir. Raporda sırasıyla; Tasarım İsterleri, Tasarım Detayları, Takım Organizasyonu ve İş Planı açıklanmaktadır.

2. Tasarım İsterleri

Yarışma kapsamında RV32IMAFB_Zicsr buyruk kümesi mimarisine sahip bir işlemci tasarlanması, bu işlemcinin tasarlanacak UART çevre birimi ile entegre bir şekilde çalışması, 4 KB boyutunda önbellek ve bu önbellekten çip dışına çıkacak ve ana belleği simüle edecek çevreleyici arayüzle bağlanması beklenmektedir. Tasarlanacak sistemlerin detaylı anlatımı Tasarım Detayları bölümünde verilmiştir.

Tasarlanacak Devre/Sistem İsmi	Tanımı ve Genel Özellikleri
RV32IMAFB_Zicsr Çekirdek (Core)	RV32IMAFB_Zicsr komut seti mimarisine göre 4 aşamalı boru hattından oluşan çekirdektir.
Bellek Arayüzü	Önbellekte olmayan verileri çip dışına taşımak için kullanılacak arayüz birimidir.
Buyruk Arayüzü	Buyrukların çip dışına çıkmasını sağlayan arayüz birimidir.
Veri Arayüzü	Verilerin çip dışına çıkmasını sağlayan arayüz birimidir.
Çevre Birimi Yöneticisi	UART destekleyen ve AXI4 Lite Bus ile çevresel birimlerin kontrolünü gerçekleştirecek olan birimdir.

Tablo 1: İşlemci İsterleri

Tahmini Performans Değerleri

Tasarımımızda ulaşılmak istenen minimum performans ölçütleri Tablo 2'de açık kaynak tasarımlar incelenerek belirlenmiştir [5] [6] [7] [8] [9] .

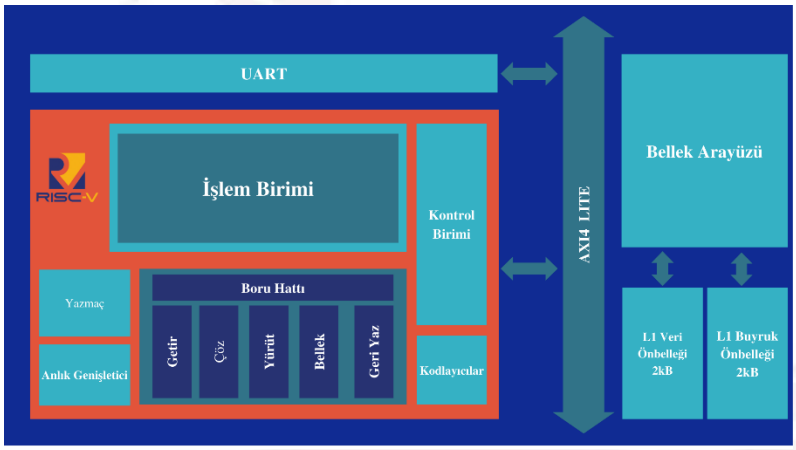
Performans Ölçütü	**Hedeflenen Minimum Performans**
Saat Frekansı	50 MHz
Coremark Skoru	Saniyede 30 İterasyon
Alan	Synopsys araçlarıyla belirlenecek minimum kapı sayısı(<1 mm^2)
Power Efficiency	<0.1
Hata Toleransı	Synopsys araçlarıyla doğrulanacak şekilde hata olmadan çalışma
Zamanlama Kapatma	Synopsys PrimeTime ile zamanlama kapatma

Tablo 2: Hedef Performans Özet Tablosu

3. Tasarım Detayları

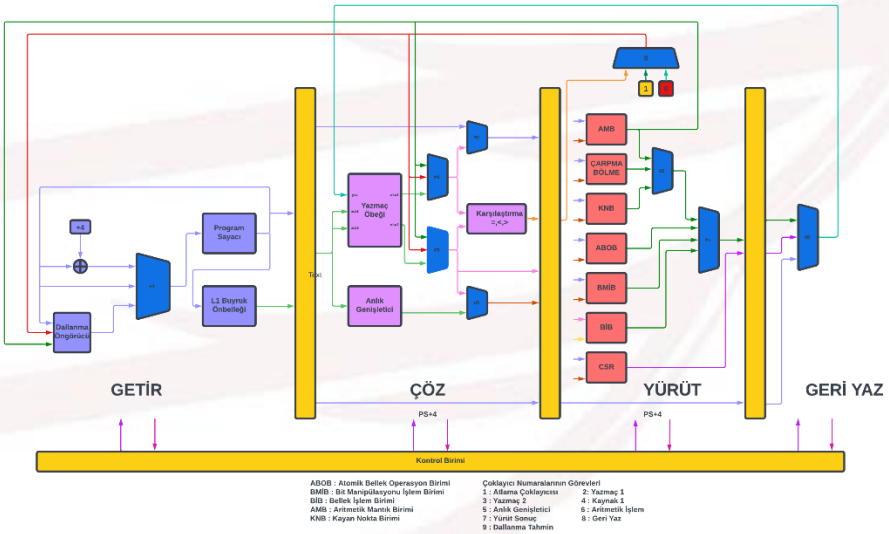
Bu bölüm yarışma kapsamında gerçeklenmesi planlanan çekirdek ve sistem detaylarını anlatmaktadır. Bölüm içeriği sistem birimleri, veri ve denetim yollarını içeren Sistem Tasarım Detayları ve boru hattı aşamalarını ve bu aşamaları gerçekleştiren birimleri içeren Çekirdek Tasarım Detayları, Bellek ve Çevresel Birim'den oluşmaktadır. Ayrıca Çip Tasarım Akışına'da bu bölümde değinilmiştir.

Aşağıda Şekil 1'de tasarlanacak işlemcinin genel sistem şeması görülebilir.



Şekil 1: Sistem Mimarisi

Aşağıda Şekil 2'de, tasarlanacak çekirdeğin boru hattı aşamaları ve işlem birimleri görülebilir.



Şekil 2: Boru Hattı

Şekil 2 incelendiğinde, planlanan işlemcinin dört temel boru hattı aşaması içerdiği görülmektedir. Tasarım sürecinde temel gerekliliklerin karşılanmasına öncelik verilmiş, bu doğrultuda açık kaynaklı işlemci mimarileri analiz edilmiş ve sonuç olarak basit ve etkili bir sıralı yürütme mekanizması içeren dört aşamalı bir işlemci tasarımı hedeflenmiştir [5] [6] [7] [8] . İşlemcinin sentez aşamasını takiben, kritik yol analizi ile boru hattının daha ince segmentlere bölünmesi mümkün olup, böylelikle performansın artırılması amacıyla tasarımın karmaşıklığının artırılması düşünülmektedir.

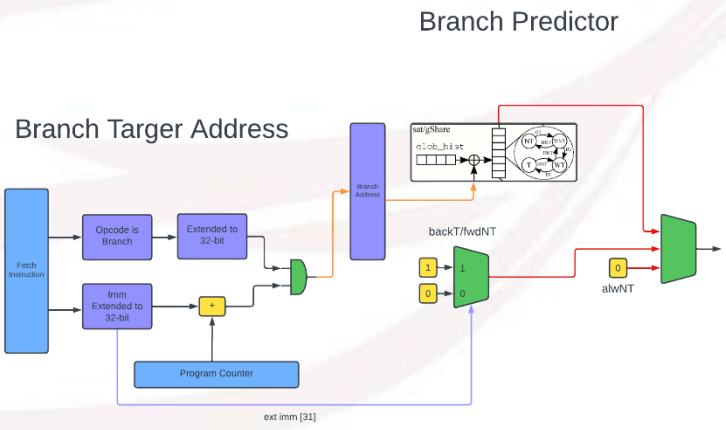
3.1 GETİR(*ing. Fetch*)

Getir: Bu aşama, program sayacının işaret ettiği konumdan buyrukların L1 buyruk önbelleğinden getirildiği aşamadır. Her çevrimde, program sayacı tarafından hesaplanan adres kullanılarak "L1 Buyruk

Önbelleği"nden uygun buyruk getirilir. Aynı zamanda bu aşama bir "Dallanma Öngörücü" birimine sahiptir, bu birim dallanma buyrukları için atlar ya da atlamaz tahminlerini gerçekleştirerek program sayacını güncelleyebilir. Program sayacının güncellenmesi, dallanma öngörücüsünün tahminine veya gerçek dallanma sonucuna bağlı olarak gerçekleştirilir.

3.1.2 Dallanma Öngörücü

Dallanma öngörüsü, mikroişlemci performansının merkezi bir parçasıdır, zira doğru yapıldığında, işlemci boru hattını verimli bir şekilde doldurur ve bu da yüksek talimat yürütme oranlarına olanak tanır [10] .Dallanma öngörüsü mekanizması dallanma tarihi tablosu (*ing. BHT*) ve program sayacı (*ing. PC*) bilgilerini kullanarak gelecekteki dallanmaların sonuçlarını tahmin eder. Bu tahminleme sürecinde başarımı arttırmak için GShare dallanma öngörücü kullanılacaktır [11]. GShare öngörücüsü, global dallanma tarihini ve Program Sayacı'nın düşük bitlerini birleştirerek dallanma tahminlerinde bulunur. Bu birleştirme, farklı Program Sayacı değerleri için ortaya çıkan benzersiz dallanma davranışlarının daha iyi eşleştirilmesine olanak tanır. GShare'in gücü, bir işlem içerisindeki farklı dallanmalar arasında ilişki kurabilme ve bu ilişkilerden yararlanarak daha hassas tahminler yapabilme yeteneğindedir. Özellikle, dallanmaların sık sık ve öngörülemez şekilde değiştiği büyük ve karmaşık işlem akışlarında, GShare dallanma öngörücüsü, daha geniş bir geçmiş yelpazesine dayanarak daha doğru tahminler sunmaktadır. GShare öngörücüsünün seçilmesinin bir diğer nedeni de koşulsuz(örn. jalr) ve koşullu(*ing. beq*) dallanmaların etkilerini daha iyi ayırt edebilmesidir. Koşullu dallanmaların sonucu öngörülebilir bir desene sahip olabilirken koşulsuz dallanmalar genellikle sabit bir davranış sergiler. GShare, bu iki tür dallanmanın geçmiş desenlerini birleştirerek ve bir 'Dallanma Geçmişi Tablosu' kullanarak daha akıllı tahminler yapabilir. Bu, özellikle birden fazla dallanmanın sıkışık olarak gerçekleştiği durumlarda önemlidir çünkü ardışık dallanmalar birbirlerinin sonuçları üzerinde doğrudan bir etkiye sahip olabilir. Ayrıca GShare, boru hattının derinleştiği ve bellek erişim sürelerinin arttığı işlemcilerde dallanma tahminlerinin hızını ve doğruluğunu artırarak önemli bir performans kazancı sağlar. Şekil 3'te GShare Dallanma Öngörücü'nün mekanizması bulunmaktadır.



Şekil 3: Dallanma Öngörücü

3.2 ÇÖZ(*ing. Decode*)

Getir aşamasından gelen buyruklar, işlemcinin "Çözücü" biriminde çözülerek mikroişlem kodlarına dönüştürülür. Bu aşamada, çözülen buyruklar işlevselliklerine göre sınıflara ayrılır ve kaynak yazma değerlerinin yazmaç öbeğinden okunması gerçekleşir. Ayrılan sınıflar, işlemci tarafından desteklenen buyruk setlerine ve işlem birimlerinin türlerine göre sınıflandırılır ve sınıflanan bu değerler, buyruk türüne ve hedeflenen işlem birimlerine bağlı olarak ilgili "Yürüt(*ing. Execute*)" aşamasındaki işlem birimlerine yönlendirilir. Bu işlem, işlemci üzerinde paralel işlem kapasitesini artırır ve verimli bir

kaynak yönetimi sağlar, böylece yürütme aşamasına geçişte her bir buyruk için gereken tüm verilerin hazır olmasını garantiler.

3.3 YÜRÜT(*ing. Execute*)

Yürüt aşaması, getirilen buyrukların işlenmesini içerir ve temel aritmetik işlemler (toplama, çıkarma, çarpma, bölme), bit düzeyinde mantıksal işlemler, dallanma kararları ve sistem çağrıları gibi bir dizi işlemi gerçekleştirir. Bu aşamada, Aritmetik Mantık Birimi (*ing. ALU*) toplama, çıkarma gibi temel aritmetik işlemleri yaparken, çarpma ve bölme işlemleri için ayrı birimler kullanılacaktır. "AFB" (*ing. Atomic, Float, Branch*) bileşeni, atomik işlemleri, kayar nokta işlemlerini ve dallanma işlemlerini yürütür. Bellek işlemleri için Yükle/Sakla birimi kullanılır ve bu birim, veri önbelleğine erişim için okuma veya yazma işlemlerini yönetir. Dallanma buyrukları, tahmin edilen sonuçlara göre boru hattı yönlendirmesini gerçekleştirir. İşlemler tamamlandıktan sonra, veri bağımlılıklarını çözmek ve performans artışı sağlamak için "Çöz" aşamasına veri yönlendirmesi yapılır bu yaklaşım verimli ve yüksek performanslı bir işlem akışı sağlamak için kritik önem taşımaktadır.

3.3.2 Aritmetik Mantık Birimi(*ing. ALU*)

Yarışma koşulları göz önünde bulundurularak, toplama, çıkarma gibi temel aritmetik işlemlerin verimli bir şekilde gerçekleştirilmesi, bir işlemcinin performansında kritik bir rol oynar. İşlemcide Aritmetik Mantık Birimi, bu temel işlemleri yüksek hızda paralel işlem yürütebilmek için özel algoritmalar ve donanımlar incelenmiştir. Hem hız hem de tasarımın serimi dikkate alınarak, toplama işlemleri için Sklansky toplayıcılar [12] kullanılmaya karar verilmiştir bu sayede toplama süresi önemli ölçüde azaltılır ve paralel hesaplama potansiyeli artırılır.

3.3.3 Çarpma/Bölme Birimi

Çarpma işlemleri toplama işlemlerine göre daha karmaşık ve kaynak yoğun olduğundan, özel algoritmalar incelenmiş alan ve performans dengesini optimize etmek için Modified Booth Dadda [13] Algoritması tercih edilmiştir. Modified Booth Dadda Algoritması, hem sayıda hem de verimde önemli iyileştirmeler sunarak, çarpma işlemlerinde kullanılan alanı minimize eder ve toplama işlemlerinden elde edilen performansı maksimize eder. Son olarak bölme işlemi, toplama ve çıkarma işlemleri kullanılarak gerçekleştirilecektir. Bunun nedeni özel bölme algoritmalarının(Non-Restoring veya SRT) getirdikleri performansa kıyasla çok yüksek bir alan gereksinimi olmasıdır.

3.3.4 Kayan Nokta Birimi(*ing. Floating Point Unit*)

Yarışma kapsamında işlemci mimarisi için tasarlanacak olan kayar nokta biriminde (-ing. FPU), performans, donanım kullanımı ve enerji verimliliği arasında optimal bir denge sağlamak amacıyla özel algoritmalar kullanılmasına karar verdik. Carry-Select Adder [14], Booth Çarpma Algoritması ve Newton-Raphson [15] bölme yöntemleri tercih edilmiştir. Carry-Select Adder, modüler yapısı ile donanım maliyetini düşük tutarken işlem hızını maksimize eder; Booth Algoritması, çarpma işlemlerini verimlileştirerek hem alan hem de enerji tüketimini optimize eder; Newton-Raphson yöntemi ise, yüksek doğrulukta ve hızlı yakınsaklıkta bölme işlemlerini sabit bir donanım alanında gerçekleştirerek güç tüketimini öngörülebilir düzeyde tutar. Bu algoritmaların entegrasyonu, işlemcimizin yüksek performans sergilemesine olanak tanırken, maliyet ve enerji verimliliği açısından da faydalı olacağı düşünülmektedir.

3.3.5 Atomik Bellek Operasyon Birimi (*ing. Atomic Memory Operation Unit*)

Yarışma kapsamında tasarlanacak Atomik Bellek Operasyon Birimi(ing. AMO) işlem birimi, çok iş parçacıklı ve eşzamanlı programlama modelleri için kritik önem taşıyan senkronizasyon mekanizmalarını verimli bir şekilde yürütebilmesi hedeflenmektedir [16] . Bu birim, 'Load-Reserved' (LR.W) ve 'Store-Conditional' (SC.W) işlemleri gibi kritik bölüm (ing. critical section) yönetimini sağlayarak, çoklu iş parçacıklı ortamlarda veri bütünlüğünü korumak için tasarlanacaktır. Ayrıca, Atomik

Bellek Operasyon Birimi talimatları - örneğin, AMOSWAP.W, AMOADD.W, vb. - veri üzerinde atomik değişiklikler yaparak hafıza erişim yarışlarını önleyecektir. Optimizasyon açısından, bu birim düşük gecikme(*ing. latency*) süreleri ve yüksek hızlı veri yolu erişimi ile tasarlanması, atomik işlemlerin verimli bir şekilde tamamlanması için gerekli olan hafıza ve işlemci kaynakları önceliklendirilecektir. Güç tüketiminin minimize edilmesi için, hafıza erişimi ve veri yolu kullanımı akıllı önbellekleme stratejileri ve veri yolunu etkili bir şekilde kullanarak optimizesi hedeflenmektedir. Ayrıca, bu işlemlerin boru hattındaki diğer talimatlarla olan etkileşimleri dikkate alınarak boru hattı engellemeleri azaltılmaya çalışılarak boru hattı verimliliği artırılması hedeflenmektedir.

3.3.6 Kontrol ve Durum Yazmacı Birimi (*ing. Control and Status Register (CSR) Unit*)

RISC-V mimarisindeki hafıza ve işlemci durumunu kontrol etmek ve takip etmek için kullanılan kontrol ve durum yazmaçları bulunur [17]. Bu yazmaçlar, işlemcinin mevcut durumunu temsil eder [18] . Bu birimde, sistem ayarlarının yönetilmesi, performans izleme, hata ayıklama, zamanlama işlemleri, güç yönetimi gibi çeşitli kritik işlevlerin etkin bir şekilde gerçekleştirilmesi hedeflenmektedir.

3.3.7 Bit-Manipülasyonu İşlem Birimi

Bu birimde, aritmetik işlemler için donanımsal olarak hızlandırılmış yollar sağlayarak, kayar nokta ve tam sayı işlemlerinde performansı iyileştirmesi hedeflenmektedir [19] . Örneğin, "Count Leading Zeros" veya "Carry-Less Multiply" gibi işlemler için özelleştirilmiş devreler incelendi. Bu devreler, genellikle çok sayıda işlem döngüsü gerektiren bit seviyesi işlemlerini tek bir döngüde tamamlayarak(örn. Barrel Shifter) işlem hızını artırmakta ve düşük güç tüketimi sağlamaktadır.

3.4 GERİ YAZ

Bu aşama, işlem birimleri tarafından tamamlanan işlemlerin sonuçlarının hedef yazmaçlara yazıldığı aşamadır. Eğer boru hattında veri bağımlılığı mevcutsa ve bu durum bir gecikmeye sebep oluyorsa, bu aşamada elde edilen değerler "Yazmaç Öbeği"ne geri yazılır ve bu, boru hattının düzgün bir şekilde akışını sürdürmesi sağlanır. Bu aşama, boru hattı içerisindeki her bir yürütme ünitesinden gelen sonuçların senkronize edilmesini sağlar ve boru hattı dengesini korur.

3.5 ANA BELLEK

Mikroişlemci mimarisinin performansında, ana bellek ve onun arayüzünün optimize edilmişi büyük bir rol oynar. Bellek hiyerarşisinin üst düzeylerinde yer alan birinci seviye önbellekler, istemci tarafından talep edilen verilerin hızlı bir şekilde sunulmasını sağlar, bu da önbellek isabet oranını artırır ve bellek erişim gecikmelerini azaltır. Bellek denetleyicisi, gelen istekleri işleyerek ve uygun bellek bloklarını önbelleğe alarak işlemleri hızlandırır. Bellekten yazma işlemlerinde, verilerin geri yazılmasını ve güncellemelerini düzenleyen write-back ve write-through politikaları devreye girer. Bellek denetleyicisi ayrıca, birden fazla istek arasında arbitraj yaparak verimliliği maksimize eder ve veri bağımlılığı durumlarında önceliği yönetir.

3.6 ÖNBELLEK

Önbellek tasarımı, sürekli erişilen verilerin hızlı ve verimli bir şekilde temin edilmesi için kritik önem taşır. Çift bağlantı noktalı(*ing. Dual-port*) tasarımı, belleğin mevcut kullanım durumunu optimize ederken, gelecekteki veri taleplerini öngörerek işlemciye hız kazandırır [20] . Belleğin genişliği, veri yolları ve önbellek boyutları, sistemin genel performansına doğrudan etki eder. Düşük gecikme süreleri ve yüksek veri transfer hızları için, özellikle 2 KB buyruk ve veri önbellekleri tasarlanır. Bu önbellekler, kontrol mekanizmaları aracılığıyla yönetilir ve güncellenir. Veri güncellenmeden önbellekten çıkarılırken, 'Write-Allocation' ve 'Write-Back' politikaları kullanılır, böylece gereksiz bellek yazmaları önlenir ve bant genişliği tasarrufu sağlanır [21] .

3.7 Çevre Birimleri

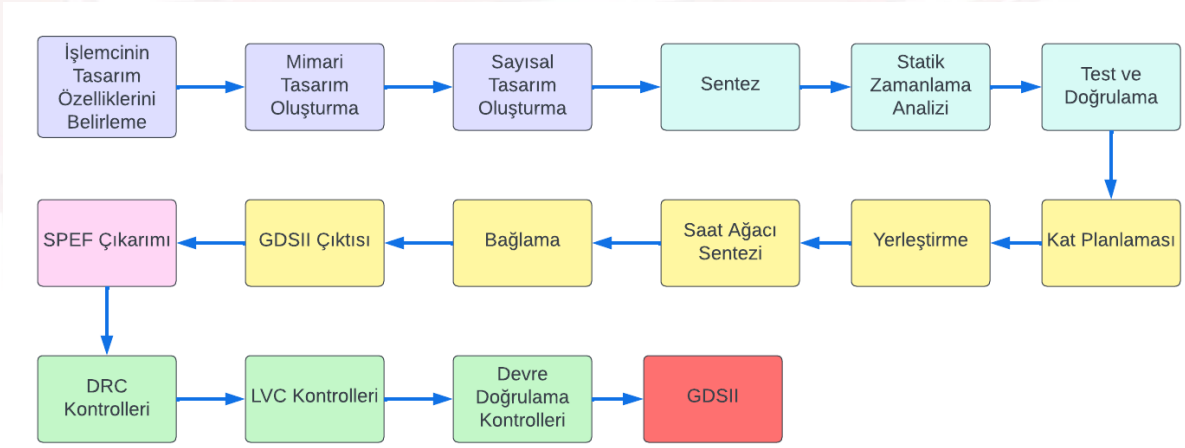
Çevre Birimleri için tasarımımızda, çekirdek veriyolu ve çevresel birimler arası iletişimde yüksek hızlı transfer ve geniş veri/adres aralıkları destekleyen AXI4-Lite protokolünü tercih ediyoruz. Bu seçim, Wishbone'a göre daha basit yapılandırma ve yüksek iletişim güvenilirliği sağlayan beş ayrı kanal yapısı sayesindedir. AXI4-Lite'ın senkron okuma/yazma işlemleri, sistem tasarımını daha öngörülebilir ve güvenilir hale getirirken, Wishbone'un asenkron işlevselliği ile karşılaştırıldığında, daha az karmaşıklık sunar ve zamanlama uyumsuzlukları riskini azaltır. Bu nedenlerle, AXI4-Lite, sistemimizin verimliliğini ve kullanılabilirliğini artırmak için stratejik bir seçimdir.

3.7.1 UART

UART modülümüz, başlangıç, veri ve durdurma bitlerini içeren seri veri paketlerinin iletimi için kullanılan standart bir iletişim protokolüdür. Bu modül, alıcı ve verici birimleri ile baud hızı üretici içerir; baud hızı üreteci, iletişim hızını ayarlar ve birimler, veri akışını yöneten pinler aracılığıyla sinyalleri işler. Entegre FIFO bellek yapıları ile donatılmış modülümüz, saat hızı farklılıklarından kaynaklanabilecek veri kayıplarını önler ve böylece verilerin AXI4-Lite veriyolu üzerinden sorunsuz bir şekilde aktarılmasını sağlar.

3.8 ÇİP TASARIM AKIŞI

Geleneksel olarak bir Uygulamaya Özgü Entegre Devre(*ing. ASIC*) tasarım sürecinin temel akışı 3 parametreden oluşur. Bunlar RTL, EDA araçları ve PDK'dir. RTL Tasarım, çipin fonksiyonel tasarımını ifade eder ve bu tasarım, EDA araçları kullanılarak geliştirilir ve doğrulanır. PDK (Process Design Kit), yarı iletken cihazların tasarımı ve üretimi için gerekli teknik bilgileri ve araçları içeren bir pakettir. Bu paket, entegre devre tasarımında kullanılan transistör modelleri, düzenleme kuralları ve malzeme özellikleri gibi çeşitli bileşen ve parametreleri kapsar. Bir ASIC tasarımın başarılı bir şekilde üretilebilmesi RTL tasarımından başlar, EDA araçları ile geliştirilip doğrulanır ve PDK ile üretime uyumlu hale getirilir. Projemizde tasarımımızın çip serimi için yarışmanın belirlediği EDA aracı ve PDK'i kullanacağız [22] .



Şekil 4: Çip Tasarım Akışı

Şekil 4'te çip tasarım akışının aşamaları yer almaktadır.

- Tasarım ve Özellik Tanımlama: Sistem gereksinimleri ve işlevselliği belirlenir, performans, güç, ve alan hedefleri dikkate alınarak VHDL veya Verilog gibi dillerle RTL tasarımı yapılır.
- Sentez: RTL tasarımı, sentez aracı kullanılarak mantık kapıları ve Flip-Flops gibi daha düşük seviyeli yapıtaşlarına dönüştürülen bir Netlist'e çevrilir.
- Statik Zamanlama Analizi (STA): Tasarımın zamanlama gereksinimlerinin karşılanıp karşılanmadığı analiz edilir.

- Tasarım Doğrulama: Tasarımın fonksiyonel doğruluğu, simülasyon, formal doğrulama ve zamanlama analizi kullanılarak kontrol edilir.
- Kat Planlaması: Çipin fiziksel layout'unun genel çerçevesi oluşturulur, güç (VDD) ve toprak (GND) bağlantılarının nasıl dağıtılacağı planlanır.
- Yerleştirme: Performans, alan kullanımı ve güç tüketimi göz önünde bulundurularak Netlist'teki bileşenler, çip üzerinde konumlara yerleştirilir.
- Clock Tree Synthesis (CTS): Saat sinyalinin, çip üzerindeki Flip-Flop'lara eş zamanlı olarak ulaşmasını sağlayacak şekilde saat dağıtım ağı uygulanır.
- Bağlama: Bileşenler arasındaki elektriksel bağlantılar çizilir, tasarımın elektriksel performansı ve sinyal bütünlüğü sağlanır.
- SPEF Extraction: Tasarımın parasitik değerleri, gerçek performansını modellemek için SPEF formatında dışa aktarılır.
- GDSII Streaming Out: Tasarım, çipin üretimi için gerekli olan GDSII formatında dışa aktarılır.
- Doğrulama: LVS ve DRC kontrolleri yapılır, fiziksel layout'un mantıksal tasarıma uygunluğu ve üretim kurallarına uygunluğu doğrulanır.

4. Takım Organizasyonu ve İş Planı

Takım organizasyonu ve iş planı bu bölümde açıklanmaktadır. Aşağıda Tablo 3'te takım organizasyonu ve özelleştirilmiş iş bölümü görülebilir. Aşağıda Tablo 4'te ise iş planı çizelgesi görülebilir. İş planımız yarışma takvimine uygun olarak ÖTR Aşaması, DTR Aşaması ve Final Aşaması olarak 3 ana aralıktan oluşmaktadır.

Takım Üyesi	Görevleri
Ahmet YOLDAŞ	Ankara Yıldırım Beyazıt Üniversitesi Elektrik Elektronik Mühendisliği 4. Sınıf öğrencisidir. Takım içi koordinasyonun sağlanması, bellek mimarisi ve bellek hiyerarşisinin araştırılması ve tasarımı, genel mimari tasarımı, çekirdek mimarisinin araştırılması ve tasarımı yapılması.
Emirhan KOCA	Orta Doğu Teknik Üniversitesi Elektrik Elektronik Mühendisliği 4. Sınıf öğrencisidir. Devre teknolojilerinin tasarlanması ve gerçeklenmesi, bellek mimarisi ve bellek hiyerarşisinin araştırılması ve tasarımı, çekirdek mimarisinin araştırılması ve tasarımı yapılması.
Sena NAL	Ankara Yıldırım Beyazıt Üniversitesi Elektrik Elektronik Mühendisliği 4. Sınıf öğrencisidir. Buyrukların araştırılması, devre tasarımı, test ve doğrulama yapılması.
Muhammet Enes AKEL	Ankara Yıldırım Beyazıt Üniversitesi Elektrik Elektronik Mühendisliği 4. Sınıf öğrencisidir. Veri yolu tasarımı, çevresel birim tasarımının gerçeklenmesi test ve doğrulama yapılması.

Tablo 3: İş dağılımı

	ÖTR AŞAMASI			DTR AŞAMASI							FİNAL AŞAMASI			
Görev Tanımı	2.01.24-5.02.24			6.02.24-15.03.24			16.03.24- 2.05.24				3.05.24-15.06.24			16.06.24-13.07.24		14.07.24-15.08.24	
Sistem Gereksinim Tespitleri ve Tasarımı																	
Ön Tasarım Raporunun Hazırlanması ve Teslimi																	
Synopsys Segmentinin Oluşturulması																	
Verilog Tasarımının Gerçekleştirilmesi																	
Ödünleşmelerin Belirlenmesi																	
Verilog Modüllerinin Testi ve Doğrulanması																	
İşlemcinin FPGA Üzerinde Demo Edilmesi																	
Detaylı Tasarım Raporunun Hazırlanması ve Teslimi																	
Oluşabilecek Hataların Revize Edilmesi																	
Tasarımın Nihai Hale Getirilmesi																	
Final Sunumunun Hazırlanması																	

Tablo 4: İş Takvimi

5. Kaynakça ve Ekler

[1] «xilinx,» [Çevrimiçi]. Available: https://www.xilinx.com/support/download.html.

[2] «kernel,» [Çevrimiçi]. Available: http://web.archive.org/web/20080207010024/http://www.808multimedia.com/winnt/kernel.html

[3] «cocotb,» [Çevrimiçi]. Available: https://www.cocotb.org.

[4] «synopsys,» [Çevrimiçi]. Available: https://www.synopsys.com/implementation-and-signoff/rtl-synthesis-test/dc-ultra.html.

[5] «opencores,» [Çevrimiçi]. Available: https://opencores.org.

[6] «openhwgroup,» [Çevrimiçi]. Available: https://github.com/openhwgroup/cva6.

[7] «rıscv,» [Çevrimiçi]. Available: https://github.com/SI-RISCV/e200_opensource.

[8] «openhwgroup,» [Çevrimiçi]. Available: https://docs.openhwgroup.org/projects/cv32e40p-user-manual/en/cv32e40p_v1.2.1/intro.html.

[9] [Çevrimiçi]. Available: https://github.com/riscv-mcu/e203_hbirdv2.

[10] S. Mittal, %1 içinde *"A survey of techniques for dynamic branch prediction,"* *Concurrency and Computation:Practice and Experience*, 2019.

[11] S. McFarling, %1 içinde *"Combining branch predictors," Citeseer, Tech. Rep*, 1993.

[12] «researhcgate,» [Çevrimiçi]. Available: https://www.researchgate.net/figure/Schematic-of-16-bit-Sklansky-adder-using-transmission-gates_fig4_258403733.

[13] S. Dod, %1 içinde *"Modified booth dadda multiplier using carry look ahead adder design and implementation,"*, 2016, p. 2229–3345.

[14] «researchgate,» [Çevrimiçi]. Available: https://www.researchgate.net/publication/338295001_Low_Power_High_Speed_Carry_Select_Adder_Design_Using_Verilog.

[15] [Çevrimiçi]. Available: https://www.researchgate.net/publication/315815658_Floating-Point_Algorithms_and_FPU_Design_in_Verilog_HDL.

[16] [Çevrimiçi]. Available: https://five-embeddev.com/riscv-user-isa-manual/Priv-v1.12/a.html#atomics.

[17] [Çevrimiçi]. Available: https://five-embeddev.com/riscv-user-isa-manual/Priv-v1.12/csr.html.

[18] [Çevrimiçi]. Available: https://ieeexplore.ieee.org/document/9677678?denied=.

[19] [Çevrimiçi]. Available: https://five-embeddev.com/riscv-user-isa-manual/Priv-v1.12/b.html#.

[20] [Çevrimiçi]. Available: https://www.sciencedirect.com/topics/computer-science/direct-mapped-cache.

[21] [Çevrimiçi]. Available: https://en.wikipedia.org/wiki/Cache_(computing)#Writing_policies.

[22] «ıcdesignflow,» [Çevrimiçi]. Available: https://ece.northeastern.edu/courses/eece4525/2018su/Lab3/IC_Design_Flow.pdf