TEKNOFEST
HAVACILIK, UZAY VE TEKNOLOJİ FESTİVALİ
ÇİP TASARIM YARIŞMASI
ÖN TASARIM RAPORU

TAKIM ADI:
KASIRGA KIZIL

PROJE ADI:
KIZIL İŞLEMCİ PROJESİ

BAŞVURU ID:
901285

11.12.2022

İÇİNDEKİLER

SEMBOLLER LİSTESİ

$	Önbellek
V$	Veri Önbelleği
B$	Buyruk Önbelleği

KISALTMALAR LİSTESİ

ASIC	Application Specific Integrated Circuit
FPGA	Field Programmable Gate Array
SRAM	Static Random Access Memory
ÖTR	Ön Tasarım Raporu
DTR	Detay Tasarım Raporu
L1	1. Seviye Önbellek

1. GİRİŞ

Bu proje kapsamında RV32IMCX buyruk kümesi mimarisine sahip RISC-V tabanlı bir işlemci tasarlanacak olup işlemci özelindeki tasarım kararları bu dokümanda açıklanmaktadır. Tasarlanacak işlemci; UART, SPI, PWM gibi çevre birimleri ve 4 KB'lik birinci seviye önbelleğe sahip olacak olup 4 aşamalı boru hattı ile işlemesi planlanmaktadır. Proje boyunca ön tasarım ve tasarım aşamalarında verilecek kararlar test edilirken Vivado [1] ve Verilator [2] gibi test ve tasarım araçlarından yardım alınacaktır. Sentez ve simülasyonlarda kullanmak üzere Yosys [3] aracının kullanımına karar verilmiştir. Sistemin tasarımı tamamlandığında, işlemci Nexys A7 XC7A100T-1CSG324C FPGA üzerinde test edilecek, OpenLane [4] ve OpenRAM [5] araçları kullanılarak sistemin ASIC fiziksel tasarımı oluşturulacaktır. Saat ağacı sentezinde ise TritonCTS [6] aracının kullanılması planlanmaktadır. Bu süreçte tüm takım üyelerinin hem FPGA üzerinde modelleme hem de ASIC tasarımı konularında tecrübe edinmesi hedeflenmektedir. Bu rapor, 3 ana bölümden oluşmaktadır. Raporda sırasıyla; Tasarım İsterleri, Tasarım Detayları, Takım Organizasyonu ve İş Planı açıklanmaktadır.

2. TASARIM İSTERLERİ

Yarışma kapsamında RV32IMCX buyruk kümesi mimarisine sahip bir işlemci tasarlanması, bu işlemcinin tasarlanacak UART, SPI, PWM çevre birimleri ile entegre bir şekilde çalışması, 4 KB boyutunda önbellek ve bu önbellekten çip dışına çıkacak ve ana belleği simüle edecek çevreleyici arayüzle bağlanması beklenmektedir.

2.1. Yürütme Şekli ve Boru Hattı Aşamaları

Çekirdek, (1)Getir, (2)Çöz ve Yazmaç Oku, (3)Yürüt ve Bellek, (4)Geri Yaz gibi üst düzeyde 4 aşamalı boru hattına sahip olacak olup sıralı (*ing.* in-order) yürütüm gerçekleştirilecektir. Boru hattının olabildiğince sade bir tasarıma sahip olması istenmiş olup işlem birimleri bazında çeşitli algoritmalar kullanılarak hızlandırma işlemlerinin gerçekleştirilmesi düşünülmektedir. Hızlandırma yöntemlerinin detayları, Bölüm 2.3 İşlem Birimleri kısmında, boru hattının detayları ise Bölüm 3.2 Çekirdek Tasarım Detayları kısmında görülebilir.

2.2. Buyruk Kümesi Mimarisi ve Özel Buyruklar

İşlemcinin buyruk kümesi mimarisi yarışma şartnamesine uygun bir şekilde RV32IMCX olarak tasarlanacaktır. Tam sayı toplama, çıkarma, çarpma ve bölme işlemlerinin desteklendiği bu buyruk kümesinin gerektirdiği işlemleri yapabilmek için yürüt aşamasında 32 bitlik aritmetik mantık birimi kullanılacaktır. Bununla birlikte, buyruk kümesinin içerdiği sıkıştırılmış buyrukları doğru bir şekilde anlamlandırmak için çöz aşaması buna göre tasarlanacaktır. Son olarak, standart RISC-V buyruk paketleri içerisinde yer almayan ve yarışma şartnamesinde işlemcinin desteklemesi gerektiği belirtilen kriptografi ve yapay zeka özel buyrukların işlenmesi için çöz aşamasında bu buyruk kodlarının anlamlandırılması ile ilgili gerekli düzenlemeler yapılacak, yürüt aşamasındaki tasarım bu özel buyrukların yürütülebileceği şekilde düzenlenecektir.

2.3. İşlem Birimleri

Aritmetik Mantık Birimi: Yarışma şartnamesinde toplama "+" ve çarpma "*" işlemlerinin davranışsal kullanılmalarına izin verilmesine rağmen tasarımımızda bu işlemler özel algoritmalar aracılığı ile gerçekleştirilecektir. Tasarım isterleri arasında OpenLane bulunması ve paralel toplayıcı algoritmalarının (*örn.* Kogge Stone [7]) alan ve serim konusundaki dezavantajlarının getirdikleri performans artılarına kıyasla yetersiz olmasından dolayı toplama algoritması olarak modifiye edilmiş carry lookahead toplayıcısı [8] kullanılacaktır. Bir diğer önemli işlem olan çarpma işlemi ise toplama işlemlerinden gerçekleştirilebilir ancak bu yöntem alandan ciddi kazanç sağlarken performans konusunda yeterli değildir. Bundan dolayı çarpma devresi Modified Booth Dadda Çarpıcısı [9] ile gerçekleştirilecektir. Bahsedilen algoritma gecikme ve performans isterlerine göre ayarlanabilir bir yapıya sahiptir. Son olarak bölme işlemi, toplama ve çıkarma işlemleri kullanılarak gerçekleştirilecektir. Bunun nedeni özel bölme algoritmalarının getirdikleri performansa kıyasla çok yüksek bir alan gereksinimi olmasıdır.

Bellek İşlem Birimi: Bu birim "Çöz - Yazmaç Oku" aşamasından gelecek olan mikroişlemlerinden yalnızca yükleme ve kaydetme mikroişlemlerinin yönlendirildiği işlem birimidir. Bu birime yönlendirilen işlemler, bu birimin ardından "Veri Önbelleği Denetleyicisi"ne yönlendirilerek Veri Önbelleğinde istenilen adresi hedef alacak şekilde yükleme ya da kaydetme işlemini gerçekleştirir.

Denetim Durum Birimi: Bu birim sistemdeki oluşabilecek olağandışı durumların (*ing.* exception) yönlendirildiği birimdir. Aynı zamanda bu durumlar oluştuğunda boru hattını boşaltabilmesine (*ing.* flush) olanak sağlaması için boru hattının büyük bir bölümüne erişimi bulunur.

Dallanma Birimi: Yürüt aşamasına gelen mikroişlemlerden dallanma başlığı altında toplananların yönlendirildiği işlem birimidir. Bu işlem biriminde istenilen dallanmanın gerçekleştirilip gerçekleştirilmeyeceği değerlendirilir. Elde edilen sonuçlar hem puanlamayı etkilemesi için "Dallanma Öngörücü"ye hem de olası yanlış bir öngörü durumunda boru hattını boşaltması için "Denetim Durum Birimi"ne yönlendirilir.

Şifreleme Birimi: Yarışma isterlerinde bulunan kriptografi ve şifreleme buyruklarının yönlendirileceği işlem birimidir.

Yapay Zeka Hızlandırıcısı: Yarışma isterlerinde bulunan konvolüsyon buyruklarının yönlendirileceği işlem birimidir.

2.4. Bellek Hiyerarşisi

Ana Bellek ve Arayüzü: Ana Bellek denetleyici, işlemcinin istediği verinin birinci seviye önbelleklerde bulunamaması durumunda devreye girer. Teknofest Komitesi tarafından sağlanan çevreleyici modülde [10] ana belleği simüle eden arayüze önbellek adresinin son 5 biti sıfırlanarak istek gönderilir ve ana bellek gecikmesi kadar çevrim sonunda istenilen 32 bitlik veri bloğu elde edilir. Aynı zamanda önbelleklerde write-back politikası sonucunda çıkarılacak blok ana bellek kontrolcüsüne yazılır ve ana belleğin giriş çıkış pinlerine bağlı olarak eş zamanlı yazma işlemi yapılır. Ana Bellek denetleyicisi bir istek ürettiğinde işlemci boru hattı durdurulacaktır. Hem (1)Getir aşaması hem de (4)Geri Yaz aşaması ana bellek isteği üretebileceği için ana bellek denetleyicisi iki isteği aynı anda kabul edip ana bellek arayüzüne bu istekleri sırayla gönderebilecektir.

Önbellek Tasarımı: Önbellekler, sık kullanılan ana bellek verilerinin kopyalarını tutan küçük ve hızlı hafıza birimleridir [11] [12]. Önbellekler, ilki alanda yerellik ve ikincisi zamanda yerellik olmak üzere iki farklı yerellik ilkesinden [13] faydalanır. Ana Bellek veri yolu genişliğinin 32 bit olması nedeniyle tasarlanacak önbellek zamanda yerellik ilkesinden faydalanamayacaktır fakat dallanma buyrukları ve sistem çağrısı (*ing.* system call) varlığında buyruk önbelleğinden faydalanılabilecektir. Önbellekler, çıkarma politikasının kolaylığı ve daha az karşılaştırıcı devre kullanması nedeniyle doğrudan eşlemeli [14] olarak gerçekleştirilecektir. Bu durum göz önüne alınarak buyruk önbelleğinin 2 KB, veri önbelleğinin ise 2 KB olmasına karar verilmiştir. Önbellekler, önbellek denetleyicileri ile kontrol edilerek "write-back" [15] olarak gerçekleştirilecektir. Bu durum, önbellekte bulunan bir verinin değeri değiştirildiği zaman ana belleğe yazılmayacak olması, bu veri önbellekten çıkarılırken değerinin ana belleğe yazılmasıdır ve "write-allocation" ile kullanılması oluşacak ana bellek isteklerini azaltacağı için başarım artışı sağlayacaktır [16] [17]. Önbelleklerin FPGA üzerinde gerçekleştirilmesi için Vivado Block Memory Generator IP'si [18] kullanılacaktır ve 130 nm OpenLane SRAM [19] serimi [4], OpenRAM aracı [5] kullanılarak çıkartılacaktır.

2.5. Dallanma Öngörüsü

Dallanma buyruklarının sonuçlarının yanlış öngörülmesi, işlemcinin öngörü yapılan aşamasından yürüt aşamasına kadar boru hattına getirilmiş buyrukların atılmasına ve performans kaybına neden olur. Mikroişlemcilerin giderek daha yüksek sayıda aşamaya sahip olması ve hafızaya erişim süresinin artması dallanma öngörüsünün doğru yapılmasının önemini arttırmaktadır. Bundan dolayı gerçeklenmesi planlanan işlemci dallanma öngörücüsü ve dallanma öngörü tablosu içerecektir. Dallanma buyrukları koşullu (*örn.* beq) ve koşulsuz (*örn.* jalr) olmak üzere ikiye ayrılabilir. Program içerisinde bu iki tür buyrukların farklı sıklıkta var olması ve farklı karakteristiğe (atlar, atlamaz) sahip olması nedeniyle bu iki tür dallanma buyrukları için iki farklı dallanma öngörü tablosu gerçeklenecektir. Koşullu dallanma öngörülerinin yönlerinin belirlenebilmesi için yüksek performanslı Gshare dallanma öngörücü [20] kullanılacaktır. Bu öngörücüde art arda iki dallanma buyruğunun geldiği durumda, arkadan gelen dallanma buyruğunun sonucunu aldığı dallanma geçmiş tablosu satırı ve sonucunu yazdığı satır farklı olmaktadır [21]. Bu problemi çözmek için dallanma geçmiş tablosu spekülatif olarak güncellenecektir.

2.6. Çevre Birimleri

Çevre birimleri, UART, SPI ve PWM olmak üzere üç birimden oluşmaktadır. Bu çevre birimleri kullanılarak farklı giriş çıkış aygıtları için farklı giriş çıkış protokolleri desteklenecektir. Yine işlemcinin programlanması, baremetal derlenmiş C kodlarının (RISC-V buyruk testleri, coremark vb.) çekirdeğe gönderilmesi bu çevre birimleri sayesinde gerçekleştirilecektir. Bu birimleri kontrol etmek için ise bellek haritalandırmalı giriş çıkış (*ing.* Memory Mapped Input Output) sistemi kullanılacaktır.

UART Kontrolcüsü: Bu birim sayesinde işlemciden gönderilen bir okuma ya da yazma isteği dış cihazlarla UART protokolünü kullanarak iletişim kurabilecektir. Kontrolcünün tasarımında incelenmek üzere birçok açık kaynaklı geçmiş tasarım araştırılmıştır. [22] [23]

SPI Kontrolcüsü: Bu birim sayesinde gönderilen bir okuma ya da yazma isteği dış cihazlarla SPI protokolünü kullanarak iletişim kurabilecektir. Birimin tasarımında datasheetler öncelikli olmak üzere SPI cihazlarıyla alakalı birçok doküman incelenmiş ve iletişimleri analiz edilerek uygun bir modül tasarlanmıştır. [24]

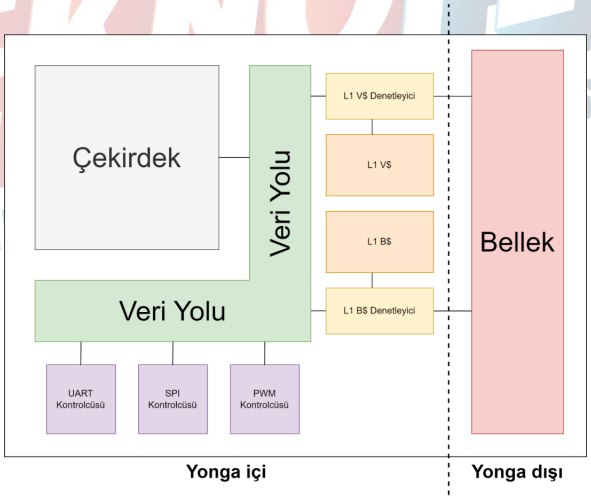
PWM Kontrolcüsü: Bu birim sayesinde gönderilen bir okuma ya da yazma isteği dış cihazlarla PWM protokolünü kullanarak iletişim kurabilecektir. Birimin tasarımında sık kullanılan mikrokontrolcülerin PWM cihazlarla nasıl iletişim kurduğu, bu mikrokontrolcülerin kütüphaneleri incelenmiş, aynı zamanda da PWM cihazları üzerinde testler yapılmıştır.

3. TASARIM DETAYLARI

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'ndan oluşmaktadır.

3.1. Sistem Tasarım Detayları

Aşağıda Şekil 3.1'de tasarlanacak yonganın genel sistem şeması görülebilir.

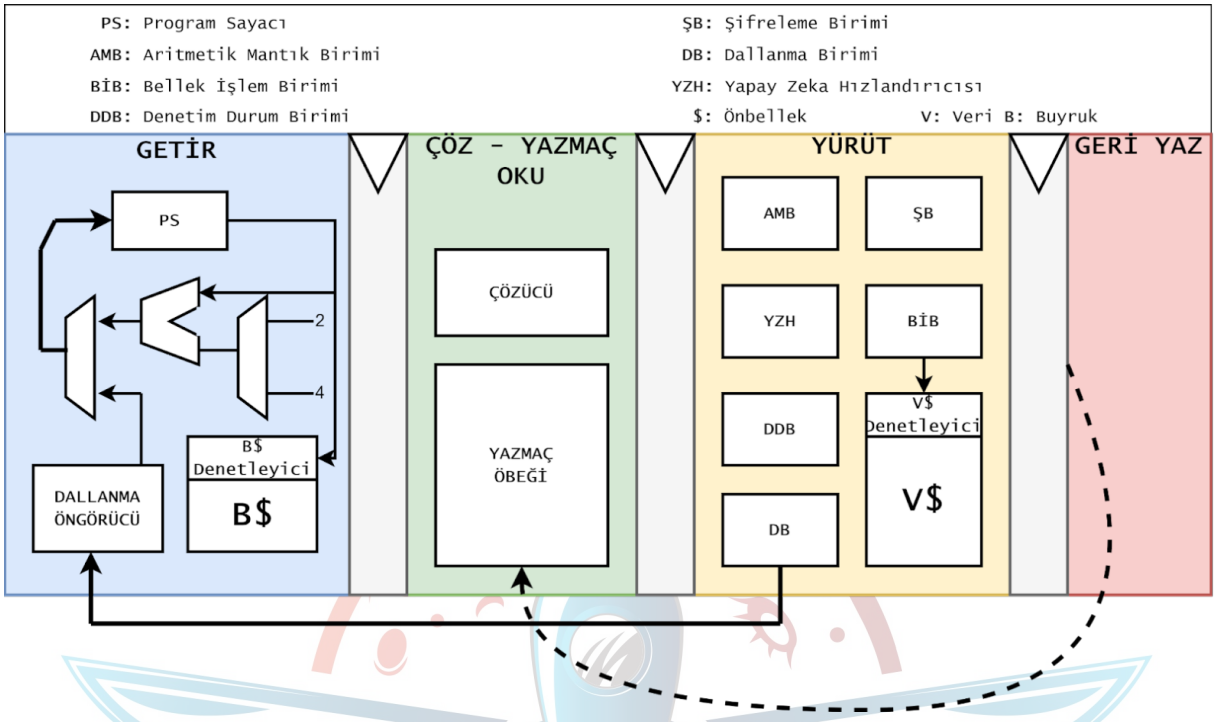


Şekil 3.1. Sistemin Genel Şeması

Şekil 3.1'den de görüldüğü üzere, sistem, buyruklara özel işlemlerin gerçekleştirildiği bir çekirdek ve çekirdeğe bir veriyolu üzerinden erişim sağlanabilen UART, SPI, PWM çevre birimleri, 2 KB L1 Veri ve 2 KB L1 Buyruk, toplam 4 KB birinci seviye önbellek ile buradan yonga dışına çıkan bellek arayüzünden oluşmaktadır.

3.2. Çekirdek Tasarım Detayları

Aşağıda Şekil 3.2'de, tasarlanacak çekirdeğin boru hattı aşamaları ve işlem birimleri görülebilir.



Şekil 3.2. Tasarlanacak Çekirdeğin Boru Hattı Şeması

Şekil 3.2'ye bakıldığında, tasarlanacak işlemcinin 4 ana boru hattı aşamasından oluştuğu görülmektedir. Tasarım kararları alınırken öncelik temel isterlerin karşılanması olduğundan basit bir tasarım seçilmiş olup açık kaynaklı işlemcilere [25] [26] [27] [28] bakılmış, ardından 4 aşamalı tasarımın sıralı yürütüme sahip bir işlemci tasarlanması planlanmıştır. OpenLane sentezinden sonra kritik yol göz önüne alınarak boru hattı alt parçalara ayrılabilir ve tasarım bundan sonra başarı iyileştirmeleri için daha karmaşıklaştırılabilir. Tasarlanacak çekirdeğin boru hattı aşamaları aşağıdaki gibi listelenebilir.

1. Getir: Program sayacının üretildiği ve bu program sayacı ile buyrukların L1 buyruk önbelleğinden getirildiği aşamadır. Bu aşamada her çevrimde hesaplanan program sayacı kullanılarak "L1 Buyruk Önbelleği"nden buyruklar okunmaktadır. Bu aşamada bir sonraki program sayacı, okunan buyruğun en anlamsız iki bitine bakılıp buyruğun sıkıştırılmış olup olmadığı kontrol edilerek belirlenir. Aynı zamanda bu aşama bir "Dallanma Öngörücü" birimine sahip olup, atlar ya da atlamaz tahminlerini kullanarak program sayacını değiştirebilecektir. Bu birim kararları alırken ise "Yürüt" aşamasındaki "Dallanma Birimi"nde geçmişte yürütülen dallanma buyruklarının sonuçlarını değerlendirmektedir.

2. Çöz - Yazmaç Oku: Getir aşamasından gelen buyrukların çözülerek mikroişlem kodlarına dönüştürüldüğü ve kaynak yazmaç değerlerinin yazmaç öbeğinden okunduğu aşamadır. Bu aşamada "Çözücü" biriminde çözülen buyruklar sınıflara ayrılır ve ayrıldıkları sınıflara göre "Yürüt" aşamasında hangi birimlere gideceğine göre mikroişlemlere ayrılır.

3. Yürüt: Buyrukların ilgili olduğu işlemlerin (aritmetik mantık, bellek, dallanma, şifreleme ve yapay zeka işlemleri) yürütülmesinin gerçekleştirildiği aşamadır. Toplama, çıkarma, çarpma, bölme ve diğer mantıksal işlemler aritmetik mantık biriminde gerçekleştirilir. Yine şifreleme işlemleri kriptografi biriminde ve yapay zeka işlemleri de yapay zeka hızlandırıcısı biriminde gerçekleştirilir. Bellek işlemlerinde ise L1 veri önbelleğine gidilerek okuma ya da yazma işlemleri yapılır. Dallanma biriminde dallanma buyruklarının atlayacağı adresler belirlenir, eğer yanlış dallanma tahmininde bulunulduysa boru hattı boşaltılır ve program sayacı güncellenir. Olağan dışı durumlarda (örneğin sıfıra bölme) CSR yazmaçları güncel-

lenip boru hattı boşaltılır. İşlemler tamamlandıktan sonra, veri bağımlılığını çözmek ve performans artışı sağlamak için Çöz - Yazmaç Oku aşamasına veri yönlendirmesi yapılır.

4. Geri Yaz: Hedef yazmacı olan buyruklar için işlem sonuçlarının bu yazmaçlara yazıldığı aşamadır. Herhangi bir şekilde veri bağımlılığı boru hattı içerisinde oluştuysa, bu aşamada elde edilen değer "Yazmaç Öbeği"ne geri yazılarak durgun haldeki boru hattını tekrar akışını sağlamak için kontrol sinyallerini düzenler.

Aşağıda Tablo 3.1'de, tasarlanacak sistem için hedeflenen minimum performans ölçütleri görülebilir. Ulaşılmak istenen minimum performans ölçütleri belirlenirken açık kaynak tasarımlar dikkate alınmıştır. [25] [26] [27] [28]

Tablo 3.1. Hedef Performans Özet Tablosu

Performans Ölçütü	**Hedeflenen Minimum Performans**
Saat Frekansı	50 MHz
Coremark Skoru	Saniyede 30 İterasyon
Alan	OpenLane'den Sorunsuz Bir Şekilde Geçmesi

3.3. Çip Tasarım Akışı

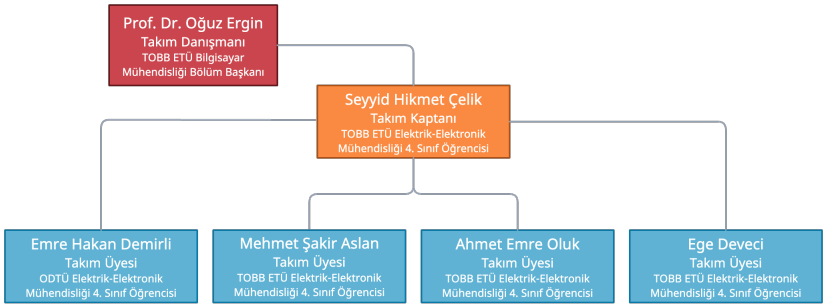
Çip Tasarım Akışında OpenLane [4] kullanılacaktır. OpenLane birçok açık kaynaklı projenin bir araya gelmesinden oluşan bir VLSI akışı projesidir. OpenLane'in RTL sentez aşamasından Yosys sorumludur. Sentezleme işleminden sonra OpenSTA tarafından kapı seviyesinde statik zamanlama doğrulaması gerçekleştirilecektir. Bu aşamada çeşitli parametreler aracılığı ile tasarım isterler doğrultusunda optimize edilebilir. Daha sonra gerçekleşecek olan Placement kısmından Floorplan, TritonMacroPlacer ve RePlAce sorumludur. Güç dağıtım ve saat ağacı ağından ise sırasıyla PDN ve TritonCTS sorumludur. Global Routing FastRoute programı aracılığı ile gerçekleştirilecek olup. Global routing çıktısı yine Yosys tarafından gerçekleştirilecek olan mantıksal eşlik testine tabi tutulur ve ardından detaylı routing işlemi TritonRoute tarafından gerçekleştirilir. Bu noktada elde edilen çıktı tekrardan akış döngüsüne sokularak çeşitli isterler doğrultusunda optimize edilebilir. Aksi takdirde OpenLane akışının kalan kısımları elde edilen fiziksel çıktının doğrulanması üzerine odaklanır. DRC, LVS, XOR eşliği ve Anten testleri gerçekleştirilir ve tasarım akışı elde edilen çıktının GDSII dosyasına çevirilmesiyle sona erer.

4. TAKIM ORGANİZASYONU ve İŞ PLANI

Takım organizasyonu ve iş planı bu bölümde açıklanmaktadır.

4.1. Takım Organizasyonu

Aşağıda Şekil 4.1'de, KASIRGA KIZIL takımının genel yapısı görülebilir.



Şekil 4.1. KASIRGA KIZIL Takımının Genel Yapısı

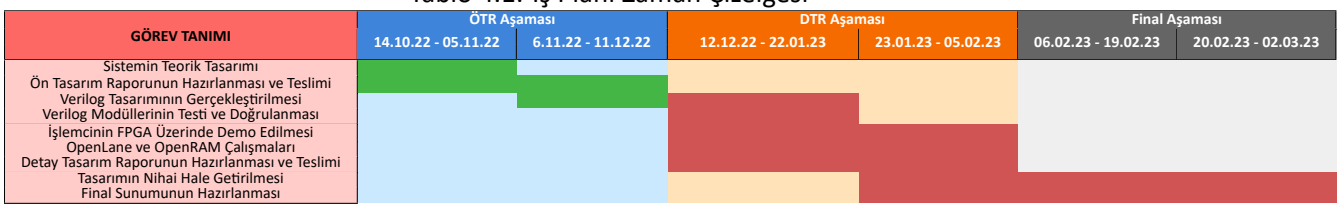
Şekil 4.1'den görüleceği üzere, 5 kişilik KASIRGA KIZIL takımı TOBB ETÜ ve ODTÜ Elektrik-Elektronik

Mühendisliği öğrencilerinden oluşmaktadır ve takımın danışmanlığını Prof. Dr. Oğuz Ergin, kaptanlığını ise Seyyid Hikmet Çelik yürütmektedir.

4.2. İş Planı

Aşağıda Tablo 4.1'de, iş planı zaman çizelgesi görülebilir.

Tablo 4.1. İş Planı Zaman Çizelgesi



Tablo 4.1'de görüldüğü üzere iş planı zaman çizelgesi yarışma tarihlerine uygun olarak ÖTR, DTR ve Final olmak üzere 3 ana fazdan oluşmakta ve her faz iki aşamaya ayrılmaktadır. Her fazın ilk aşaması o fazın şartlarını yerine getirmek için olan hazırlık aşamasını kapsar.

Aşağıda Tablo 4.2'de ise kişiler bazında daha özelleştirilmiş iş bölümü görülebilir.

Tablo 4.2. Takım Üyeleri İş Bölümü

Takım Üyesi	Atanan Görevler
Seyyid Hikmet Çelik	Takım Organizasyonunun Sağlanması Verilog Tasarımı: Çöz - Yazmaç Oku, Ana Çekirdek Modülü C Kodlarının Derlenmesi, Yürütümü ve Testi
Emre Hakan Demirli	Verilog Tasarımı: Denetim Durum Birimi, PWM Modülleri Tasarım Doğrulama OpenLane ve OpenRAM
Mehmet Şakir Aslan	Verilog Tasarımı: Önbellek Denetleyicileri, Bellek İşlem Birimi, Dallanma Öngörücüsü, Getir ve SPI Modülleri
Ahmet Emre Oluk	Verilog Tasarımı: Şifreleme ve Yapay Zeka Buyrukları, Geri Yaz aşaması Boru hattının doğrulanması ve geliştirilmesi OpenLane ve OpenRAM
Ege Deveci	Verilog Tasarımı: Dallanma, Aritmetik Mantık Birimi ve UART modülleri Tasarım Doğrulama

KAYNAKÇA

1. https://en.wikipedia.org/wiki/Xilinx_Vivado.
2. http://web.archive.org/web/20080207010024/http://www.808multimedia.com/winnt/kernel.htm.
3. https://github.com/YosysHQ/yosys.
4. https://github.com/The-OpenROAD-Project/OpenLane.
5. https://openram.org.
6. https://github.com/The-OpenROAD-Project/TritonCTS.

7. https://en.wikipedia.org/wiki/Kogge–Stone_adder.

8. https://en.wikipedia.org/wiki/Carry-lookahead_adder.

9. S. Dod, “Modified booth dadda multiplier using carry look ahead adder design and implementation,” *International Journal of Computer Science & Engineering Technology (IJCSET), ISSN*, pp. 2229–3345, 2016.

10. https://github.com/TUTEL-TUBITAK/TEKNOFEST_2023_Cip_Tasarim_Yarismasi/tree/main/Wrapper.

11. A. J. Smith, “Cache memories,” *ACM Computing Surveys (CSUR)*, vol. 14, no. 3, pp. 473–530, 1982.

12. R. Sancheti and K. Ramesh, “Cache design and optimization techniques,” *Research and Applications: Emerging Technologies*, vol. 3, no. 3, 2022.

13. https://en.wikipedia.org/wiki/Locality_of_reference.

14. https://www.sciencedirect.com/topics/computer-science/direct-mapped-cache.

15. https://en.wikipedia.org/wiki/Cache_(computing)#Writing_policies.

16. https://www.geeksforgeeks.org/write-through-and-write-back-in-cache.

17. https://www.techtarget.com/whatis/definition/write-back.

18. https://docs.xilinx.com/v/u/en-US/pg058-blk-mem-gen.

19. https://github.com/efabless/sky130_sram_macros.

20. M. Evers and T.-Y. Yeh, “Understanding branches and designing branch predictors for high-performance microprocessors,” *Proceedings of the IEEE*, vol. 89, no. 11, pp. 1610–1620, 2001.

21. I. Kim, J. Jun, Y. Na, and S. W. Kim, “Design of a g-share branch predictor for eisc processor,” *IEIE Transactions on Smart Processing and Computing*, vol. 4, no. 5, pp. 366–370, 2015.

22. https://github.com/hell03end/verilog-uart.

23. https://github.com/ZipCPU/wbuart32.

24. https://zipcpu.com/blog/2018/08/16/spiflash.html.

25. https://github.com/openhwgroup/cva6.

26. https://github.com/openhwgroup/cv32e40p.

27. https://github.com/chipsalliance/Cores-SweRV.

28. https://github.com/SI-RISCV/e200_opensource.