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

TAKIM ADI

KASIRGA

PROJE ADI

KASIRGA İŞLEMCİ PROJESİ

BAŞVURU ID

469696

10.07.2022

İÇİNDEKİLER

1. TEMEL TASARIM ÖZETİ . . . 1
2. PROJE MEVCUT DURUM DEĞERLENDİRMESİ . . . 1
3. PROJE DETAY TASARIMI . . . 2
   3.1. Sistem Mimarisi . . . 2
   3.2. Tasarım Detayı . . . 3
      3.2.1. Çekirdek Tasarımı . . . 3
         3.2.1.1. Ön Taraf . . . 3
         3.2.1.2. Arka Taraf . . . 6
         3.2.1.3. Denetim Durum Birimi . . . 8
      3.2.2. Bellek Tasarımları . . . 8
         3.2.2.1. Önbellekler ve Arayüzleri . . . 9
         3.2.2.2. Ana Bellek Arayüzü . . . 10
      3.2.3. Çevre Birimleri Tasarımları . . . 11
         3.2.3.1. Adres Tekleyici . . . 11
         3.2.3.2. UART . . . 12
         3.2.3.3. SPI . . . 12
4. ÇİP TASARIM AKIŞI . . . 12
   4.1. Makro Tasarım Akışı . . . 12
   4.2. SRAM Tasarım Akışı . . . 13
   4.3. İşlemci Tasarım Akışı . . . 14
5. TEST . . . 15
   5.1. RTL Seviyesindeki Simulasyon Ortamı . . . 15
   5.2. FPGA Test Ortamı . . . 16
   5.3. Serim Sonrası Kapı Seviyesi Test Ortamı . . . 16
6. TAKIM ORGANİZASYONU . . . 16
   6.1. Takım Organizasyonu . . . 16
   6.2. Görev Dağılımı . . . 16
7. İŞ PLANI ve RİSK PLANLAMASI . . . 17
KAYNAKÇA . . . 18

1. TEMEL TASARIM ÖZETİ

Bu rapor, proje kapsamında geliştirilen işlemcinin mimari tanımlamasını, tasarımını, çip tasarım akışını ve FPGA ortamındaki gerçekleme ve doğrulama akışını içermektedir.İşlemci çekirdeği, proje isteri olarak belirlenen RV64IMCX buyruk kümesi mimarisini desteklemektedir. Yarışma kapsamında gerçeklenen çekirdek tasarımı sıralı yürütüm ile buyrukları işlemektedir. Çekirdek, 6 boru hattı aşamasından oluşmaktadır: (1) Getir1 *(-ing. fetch)*, (2) Getir2, (3) Çöz *(-ing. decode)*, (4) Yazmaç Oku *(-ing. register read)*, (5) Yürüt *(-ing. execute)*, (6) Geri Yaz *(-ing. write-back)*.

İşlemcinin performansını artırmak için çekirdekten yapılan bellek istekleri için proje istelerinden biri olan iki seviyeli önbellek yapısı kullanılmaktadır. Veri ve buyruk için ayrı birinci seviye önbellekler kullanılmaktadır. Her iki önbellek de doğrudan eşlemeli 128 satırlı ve 128-bit veri öbeği genişliğine sahip olup 2KB boyutundadır. İkinci seviye önbellek paylaşımlı ve 8 KB 4 yollu kümeli ilişkilidir. Bellek hiyerarşisinin son adımı olan özelleştirilmiş ana bellek denetleyicisi ile çip dışındaki bellek ile haberleşilir. Proje isterlerinden olan SBox kullanılarak verileri ana bellekte şifreleyerek saklama ve çözerek çekirdeğe aktarma işlemi ana bellek denetleyicisi tarafından gerçekleştirilmektedir. İşlemci, çevre birimleri olarak UART ve SPI protokollerini desteklemektedir. İşlemci, çip dışıyla haberleşmesini ana bellek denetleyicisine ek olarak, çevre birimleri olan UART ve SPI ile gerçekleştirmektedir.

Çipin serimini gerçekleştirmek için OpenLane (Ghazy & Shalan, 2020) akışı kullanılmıştır. Bu akışta her bir bileşen (çekirdek, çevre birimleri, önbellek denetleyicileri, vb.) ayrı olarak makrolaştırılmıştır. Çip seriminde önbellekler yazmaç tabanlı bellek yerine SRAM tabanlı bellek olarak gerçeklenmiştir. SRAM tabanlı bellekleri çip akışında makrolaştırmak için OpenRAM (Guthaus et al., 2016) aracı kullanılmıştır. Elde edilen makrolar, genel bir birim altında bağlanarak işlemcinin serimi elde edilmiştir.

Sistemin davranışsal işlevselliği, hem simulasyon ortamında hem de 125MHz saat sıklığında gerçeklenen FPGA tasarımı üzerinde RISC-V testleri (RISC-V, 2010) yürütülerek doğrulanmıştır. RISC-V testlerine ek olarak çeşitli senaryoları test eden mikro programlar yazılmış ve çalıştırılmıştır.

2. PROJE MEVCUT DURUM DEĞERLENDİRMESİ

Proje kapsamında geliştirilen işlemci çekirdeği ön tasarım raporunda da belirtilen bütün isterleri karşılamaktadır. RV64IMCX buyruk kümesi mimarisini *(-ing. instruction set architecture)* destekleyen işlemci gerekli bütün testleri başarıyla geçmiştir. Projenin ilerlemesi sırasında ön tasarım raporunda belirtilenden farklı şekilde uygulanan bazı tasarım kararları alınmıştır.

- Çarpma ve bölme işlemlerindeki gecikmeyi azaltmak için ilgili birimlerde Kogge-Stone tabanlı toplama ve çıkarma işlemi yerine artı ve eksi operatörleri kullanılarak tasarım gerçeklenmiştir. Toplama ve çıkarma işlemleri için ise aritmetik mantık biriminde Kogge-Stone algoritması kullanılmaya devam edilmiştir.
- Alanda yerellik ilkesinden faydalanarak sistemin performanısı arttırmak adına önbelleğin

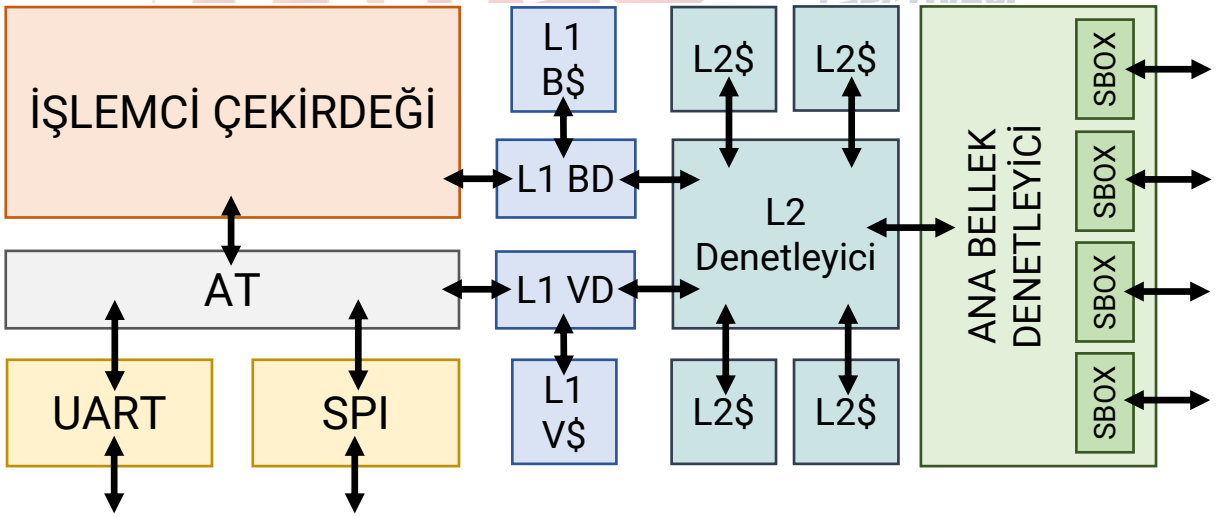
veri öbeği boyutu 32-bitten 128-bite çıkarılmıştır. 1. seviye önbellekler ve 2. seviye önbelleğin her yolu, her satırı 32-bit olan 512 satırlı değil her satırı 128-bit olan 128 satırlı olacak şekilde gerçekleştirilmiştir.

3. PROJE DETAY TASARIMI

3.1. Sistem Mimarisi

Proje kapsamında geliştirilen sistemin genel blok şeması Şekil 3.1'de gösterilmektedir. Boru hattı ve sistem genelinde herhangi bir açık kaynak tasarım kullanılmamış ve çekirdek tamamen özgün bir şekilde gerçeklenmiştir. Sistemde 6 aşamalı sıralı yürütme yapan işlemci çekirdeği, işlemci çekirdeğinin Getir1 ve Getir2 aşamalarıyla haberleşen 2 KB boyutunda 1. seviye buyruk önbelleği (L1B$) ve denetleyicisi (L1 BD), yürüt aşamasındaki bellek işlem birimiyle haberleşen 2 KB boyutunda 1. seviye veri önbelleği (L1V$) ve denetleyicisi (L1 VD), 1. seviye önbelleklerde olmayan veriler için 8KB boyutunda bir adet paylaşımlı 2. seviye önbellek (L2$) ve denetleyicisi, bir adet çip dışı bellek birimiyle haberleşmek ve ana bellekte verileri karıştırılmış bir şekilde tutulmasını sağlamak için ana bellek denetleyici, çekirdek dışı birimler (çevre birimleri ve bellek hiyerarşisi) ile çekirdeğin haberleşmesini yönelndirmek için adres tekleyici (AT) ve çip dışıyla haberleşmeyi gerçekleştirmek için UART ve SPI çevre birimleri yarışma şartnamesinde istenildiği şekilde özelleştirilerek sisteme entegre edilmiştir. Sistem mimarisinin yazmaç seviyesindeki kodlaması Verilog-2005 donanım tasarlama dili kullanarak gerçeklenmiştir.

İşlemci çekirdeği, RV64IMCX özelleştirilmiş buyruk kümesi mimarisini desteklemektedir. Çekirdekte toplama işlemi, toplama sembolü yerine Kogge-Stone (Kogge & Stone, 1973) algoritması kullanılarak gerçekleştirilmiştir. Çarpma işleminde çarpma sembolü yerine Booth algoritması (Booth, 1951), bölme işleminde bölme sembolü yerine yeniden kaydetmeli bölme algoritması *(-ing restoring division algorithm)* (Takagi, Kadowaki, & Takagi, 2005) kullanılmıştır.



Şekil 3.1: Sistem Blok Şeması.

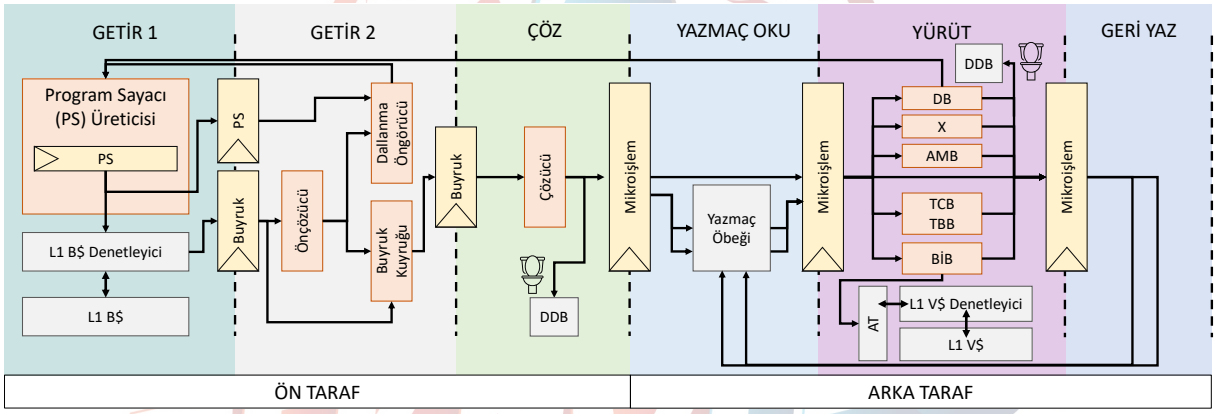
3.2. Tasarım Detayı

Bu bölümde gerçeklenen sistemin bileşenleri ve tasarım detayları ayrıntılı bir şekilde açıklanmaktadır. Sistem üç ana bölümden oluşmaktadır: (i) Çekirdek, (ii) Bellek Hiyerarşisi ve (iii) Çevre Birimleri.

3.2.1. Çekirdek Tasarımı

İşlemci çekirdeği sıralı yürütüm (*-ing. in-order execution*) gerçekleştirmektedir ve 6 aşamalı boru hattına sahiptir. Şekil 3.2 çekirdek tasarımına ait blok şemasını göstermektedir. Boru hattı boyunca kritik yolu oluşturan işlemlerin gerektirdiği kaynaklar göz önünde bulundurularak boru hattı 6 aşamaya ayrılmıştır. Bu aşamaları aşağıda üç ana başlık altında açıklanmaktadır.

3.2.1.1. Ön Taraf. Ön taraf (*-ing. front-end*) Getir1, Getir2 ve Çöz aşamalarından oluşmaktadır ve buyrukların boru hattına getirilmesi ve çözülmesinden sorumludur. Ön taraf, program akışına uygun buyrukları boru hattına getirerek program sayacını belirlemekte ve buyrukların işlendiği arka taraf aşamalarına buyrukları mikroişlemlere çevirerek iletmektedir.



Şekil 3.2: Çekirdek Blok Şeması.

Getir1. Boru hattının ilk aşamasıdır. Bu aşamada, program akışına uygun olarak program sayacı üretilmekte ve bu program sayacı ile ilk seviye buyruk önbelleğine erişilerek buyruklar getirilmektedir. Boru hattına tek seferde 32 bit alınmaktadır.

Program sayacının belirlenmesinde etkili olan iki durum bulunmaktadır. İlk olarak, program akışı değişmediği takdirde program sayacı her çevrim 4 artırılır ve bu sayede bir sonraki buyruğu boru hattına getirir. İkinci olarak program akışının değişmesi ile program sayacı farklı değerler ile güncellenmelidir. Program akışının değişmesi üç durumda gerçekleşmektedir. İlk olarak Getir2 aşamasında gerçekleşen öncöz (*-ing predecode*) sonucunda program akışını değiştiren buyrukların (dallanma ve atlama buyruklarının) tespiti ile Getir2 aşaması program sayacını güncelleyebilmektedir. İkinci olarak Yürüt aşaması, öngörülen program akışının doğrulanması sırasında getirilen buyrukların doğru olup olmadığına karar vermektedir. Bu aşamada, yapılan öngörünün yanlış olduğu tespit edilirse program akışının düzeltilmesi için program sayacı doğru değerle güncellenmelidir. Üçüncü olarak ise olağandışı

durum gerçekleştiği takdirde program sayacı bu olağan dışı durumun çözülebilmesi için program sayacını güncelleyerek akışı değiştirebilmektedir. Bu üç durumdan olağandışı durum oluşması diğerlerinden, Yürüt aşamasından gelen güncelleme ise Getir2 aşamasında gelen güncellemeden daha önceliklidir. Bunun sebebi boru hattında daha ileride bulunan buyrukların getirdiği güncellemelerin program akışını değiştirme sırasında daha önce gelmesidir.

Getir2. Boru hattının ikinci aşamasıdır ve getirilen buyruğun önçözümünün yapılmasından ve dallanma öngörüsünden sorumludur. Sıkıştırılmış (*-ing. compressed*) buyrukların desteklenmesi için bu aşamada bir durum makinesinden yararlanılmıştır. Getirilen buyrukların sıkıştırılmış ya da standart buyruk olması durumları göz önünde bulundurularak buyruklar bir buyruk kuyruğu aracılığı ile sonraki aşama olan Çöz aşamasına gönderilmektedir. Sıkıştırılmış buyruklar sebebi ile buyruk adreslerinin dördün katı olmaması sonucunda buyruk kuyruğuna eklenen buyrukların program sayacı değerleri de bu aşamada kontrol edilmekte ve doğru bir biçimde Çöz aşamasına iletilmektedir.

Sıkıştırılmış Buyrukların Desteklenmesi. Getirilen 32 bitin iki yarısında sıkıştırılmış buyruk işlem kodlarına bakılarak buyrukların sıkıştırılmış olup olmadığına karar verilmektedir. Burada üç farklı durumdan söz edilebilir. İlk olarak, getirilen 32 bitin standart boy bir buyruk olması durumunda buyruk direkt olarak Çöz aşamasına iletilmekte ve program sayacı aynı şekilde kullanılmaktadır. İkinci olarak getirilen 32 bit iki ayrı sıkıştırılmış buyruktan oluşabilmektedir. Bu durumda ilk buyruk ve program sayacı Çöz aşamasına iletilmekte ve ikinci buyruk buyruk kuyruğuna yazılmaktadır. Bu durumda Getir1 aşaması bir sonraki çevrim buyruk göndermemesi için duraklatılmaktadır. Bir sonraki çevrim, buyruk kuyruğuna yazılmış olan ikinci buyruk Çöz aşamasına iletilmekte ve program sayacı 2 artırılmaktadır. Üçüncü durum, gelen ilk buyruğun sıkıştırılmış ve ikinci buyruğun ise standart buyruk olması durumudur. Bu durumda ilk buyruk Çöz aşamasına program sayacı değeri ile birlikte gönderilmekte ve 32 bitin ikinci yarısı buyruk kuyruğuna eklenmektedir. Bir sonraki çevrimde Getir1'den alınan 32 bitin ilk yarısı, kuyruktaki 16 bite eklenerek ikinci buyruk tamamlanmaktadır. İkinci buyruğa ait program sayacı, buyruğun ilk yarısı ile gelen program sayacının 2 artırılması ile elde edilmektedir. Bir buyruğun 32 bitin ilk 16 bitinde tamamlanması durumunda iki durum ortaya çıkmaktadır. İkinci yarıda bir sıkıştırılmış buyruk varsa bu buyruk kuyruğa eklenmekte ve Getir1 aşaması duraklatılmaktadır. Bu durumda, program sayacı 2 artırılarak bir sonraki çevrim sıkıştırılmış buyruk Çöz aşamasına gönderilmektedir. İkinci olarak ise 32 bitin son 16 bitinin yine yarım bir standart buyruk olmasıdır. Bu durumda tekrar bir sonraki 32 bitin getirilmesi beklenmekte ve durum makinesi yukarıda açıklandığı şekilde işlemeye devam etmektedir.

Sıkıştırılmış atlama buyrukları sebebi ile, program sayacı dördün katı olmayan adreslere atlayabilmektedir. Bu durum Getir2 aşamasında program sayacının denetlenmesi ile tespit edilerek adresi dördün katı olacak şekilde getirilen buyruğun doğru yarısının buyruk olarak değerlendirilmesi sağlanmaktadır.

Önçöz. Çöz aşamasına iletilecek buyruk belirlendiğinde bu buyruğun program akışını değiştiren bir buyruk olması durumu kontrol edilmektedir. Program akışını değiştiren buyruklar dallanma ve atlama buyruklarıdır. Bu buyruklara ait işlem kodları tespit edildiğinde buyruklar işaretlenmekte ve dallanma biriminden bu buyruklar için atlama öngörüsü ve adresi

alınmaktadır.

Dallanma Öngörü Birimi. Bir dallanma buyruğunun doğru şekilde öngörülebilmesi için bu dallanma buyruğunun sonuç ve dallanma adresinin tahmin edilmesi gerekmektedir. Dallanma sonucunun tahmin edilmesi ÖTR'de de açıklandığı üzere GShare (Gao & Zhou, 2005) öngörücünün kullanılması ile gerçekleştirilmektedir. GShare öngörücü, genel geçmiş ile yerel geçmişi bir arada kullanarak yüksek başarımla dallanma öngörüsünde bulunan bir mekanizmadır. Öngörüler için çift kutuplu sayaçlardan oluşan bir dallanma öngörü tablosundan yararlanmaktadır. Bu tabloya 5 bitlik genel dallanma geçmişinin tutulduğu yazmaç ve program sayacının XOR işlemine tabi tutulması ile oluşturulan bir indis ile erişilmektedir. Genel geçmişin tutulması sayesinde farklı örüntüye sahip ve tekrar eden dallanma buyrukları farklı tablo satırlarına erişmekte ve farklı öngörülerin yapılması sağlanmaktadır.

Dallanma buyruklarının dallanma adreslerinin tahmin edilmesi Dallanma Hedef Arabelleği (*-ing. Branch Target Buffer veya BTB*) ile gerçekleştirilmektedir. Tasarlanan BTB, GShare tablosunun bir uzantısı gibi çalışmaktadır ve GShare erişim adresi ile dallanma adreslerine erişmektedir.

Dallanma buyruğunun sonucu Yürüt aşamasındaki Dallanma Birimi tarafından hesaplanmaktadır. Dallanma birimi, öngörücüde yapılan dallanma öngörüsü ve öngörülen dallanma adresini öngörülen program akışı ile karşılaştırmaktadır. Dallanma sonucu belirlendiğinde Dallanma Öngörücü bu sonuca bağlı olarak güncellenmektedir. Dallanma buyruğunun program sayacı ve geldiği zamanki genel geçmiş yazmacı değeri kullanılarak öngörü için kullanılan çift kutuplu sayaca erişilmektedir. Sayaç, dallanma atladı ise artırılmakta, atlamadı ise azaltılmaktadır. Bu sayede değişen program akışlarına adaptasyon sağlanmaktadır. Yapılan öngörünün yanlış olması durumunda, dallanma öngörüsü sonrasında boru hattına getirilen buyruklar geçersiz kılınmakta ve program sayacı program akışına uygun buyruk adresi ile güncellenmektedir.

Çöz. Tasarlanan Çöz aşamasında hedeflenen buyrukların Getir1 ve Getir2 aşamalarından program sırasına uygun olarak getirilmesinden sonra sınıflandırılması ve boru hattının geri kalanında kullanılacak mikroişlem sinyallerinin doğru bir şekilde üretilmesinin sağlanmasıdır. Çöz aşamasının oluşturduğu mikroişlem sinyalleri ile diğer boru hattı aşamaları, boru hattı içerisindeki buyruğun ne olduğu, buyruğun işlenmesi için hangi verilerin ve yazmaçların gerekli olduğu, bu yazmaçların hangi şekilde işleme sokulması gerektiği gibi birçok önemli bilgiye ulaşabilir. Tasarlanan mikroişlemde buyruğun hangi birimde işleneceğini ve bu birim içinde uygulanacak işlemin ayırt edilmesini sağlayan mikroişlem kodları yürüt kodu ve işlev kodu olarak isimlendirilmiştir. Yürüt kodu, yürüt aşaması içerisindeki alt birimler arasında hangisinin seçileceğine belirtir. Bu alt birimler yürütülen buyruğun türüne ve özelliklerine göre AMB (Aritmetik mantık birimi), DB (Dallanma birimi), BİB (Bellek işlem birimi), TCB (Tamsayı çarşamba birimi), TBB (Tamsayı bölme birimi), DDB (Denetim durum birimi), XB (X birimi) olabilir. İşlev kodu ise seçilen alt birimler içerisinde buyruğun hangi işlemi gerçekleştireceği ile ilgilidir. Örneğin, AMB'de işlem yapılmasını gerektiren bir buyruğun işlev koduna göre yapılacak aritmetik işlemin (çıkarma, toplama, kaydırma vb.) türü belirlenir. Aynı zamanda yürütülen buyruğa uygun sistem kaynaklarının mikroişlem sinyalleri üzerinden belirlenmesi de Çöz aşamasında sağlanır.

RISC-V buyrukları belirlenen görevleri yerine getirebilmek için yazmaçları ve anlık değerleri kullanır. Buyruğun türüne ve buyruk mimarisi spesifikasyonundaki haritalandırmasına göre çeşitli anlık ve yazmaç türleri sunulmuştur. Çöz içerisinde buyruğun işlem kodu (*-ing. op-code*) ve fonksiyon değerleri ayrıştırılarak, hangi anlık tipinin ve yazmaç konfigürasyonun kullanılacağı belirlenmektedir. Bu sayede varsa kaynak ve hedef yazmaçları ile birlikte anlık değeri, buyruğun işleneceği alt birime mikroişlem sinyalleri sayesinde iletilir. RV64 buyruk mimarisinin desteklenmesi ile birlikte sistemde 64 bitlik veri ile işlem yapabilen buyruklar oluşmaktadır.

Tasarlanan Çöz aşaması buyruk için kullanılması gereken sistem kaynaklarını işleç isimli mikroişlem sinyalleri ile sonraki aşamalara iletir. Bunun yanında RV64 desteğinin gelmesi ile birlikte çift kelimeli (*-ing. double word*) yani 64 bitlik işlemlerin ayırt edilebilmesi için de mikroişlemde bir bitlik sinyal belirlenmiştir. Bellek işlemlerinin ayırt edilmesi ve bellek işleminin işaretle genişletilmesi gerekliliği bilgisi de mikroişlem sinyalleri ile Çöz aşamasında atanmaktadır. Sıkıştırılmış buyruklarının desteklenebilmesi için Çöz aşaması gelen buyrukların ilk 2 bitine bakarak buyruğun sıkıştırılmış olup olmadığına karar verir ve sıkıştırılmış ise Çöz aşamasının içerisinde genişletme işlemi yaparak sıkıştırılan buyrukları standart boyutta buyruklara dönüştürür. Daha sonra ise buyruklar çözümlenmektedir. Getir2 aşamasında sıkıştırılmış buyruklar her çevrimde tek buyruk olacak şekilde Çöz aşamasına iletilmektedir. Bu nedenle bu aşamada yalnızca tek buyruk çözülür. X buyruklarının desteklenmesi için şartnamede sunulan buyruk haritalandırmasına uygun çözme işlemleri uygulanmaktadır.

3.2.1.2. Arka Taraf. Arka taraf Yazmaç Oku, Yürüt ve Geri Yaz aşamalarından oluşmaktadır ve Ön tarafta oluşturulan mikroişlemlerin doğru bir şekilde yürütülmesinden sorumludur. Arka taraf, geçerli olan mikroişlemler için gerekli yazmaçlar hesaplayarak sonuçları ilgili bellek adresine ya da yazmaçlara yazar. Ek olarak, dallanma öngörücünün öngörüsünü kontrol edip program akışını yanlış ilerlemesini engeller.

Yazmaç Oku. Boru hattının dördüncü aşamasıdır. Bu aşamada, Çöz aşamasında oluşturulan mikroişlemlere bakılarak Yürüt aşamasında kullanılacak yazmaçlar yazmaç öbeğinden (*-ing. register file*) okunur. Okunan yazmaçların değerleri mikroişleme kaydedilir ve bir sonraki boru hattı aşamasına iletilir. Yazmaç öbeğinden okuma işlemi yapılırken dikkat edilmesi gereken bazı durumlar vardır. Bu durumlar (1) yazma-sonrası-okuma (*-ing. read-after-write*) ve (2) yazma-sonrası-yazma (*-ing. write-after-write*) veri bağımlılıklarıdır.

Bu veri bağımlılıkları, çalıştırılan programın doğru bir şekilde işlemesi için gereklidir. Şekil 3.3'deki örnek kod parçacığı, dikkate alınmadığı durumda yanlış işleyebilecek bir programdır. İlk iki buyruk x1 yazmacına yazma, üçüncü buyruk ise x1 yazmacına okuma yapmaktadır. Üçüncü buyruk Yazmaç Oku aşamasındayken, ikinci ve ilk buyruklar sırasıyla Yürüt ve Geri Yaz aşamasındadır. Programın doğru işlemesi için, üçüncü buyruk x1 yazmaç değeri olarak, ikinci buyruğun sonucunu almalıdır. Bu sorunu ya veri yönlendirmesi ya da bekleme yöntemi *(-ing, stalling)* ile çözülebilir. Veri yönlendirmesi tekniği programın daha hızlı ilerlemesini sağlamasına rağmen boru hattı aşamaları arasındaki sinyalleri ve tasarım karmaşıklığını arttıracağından tercih edilmemiştir. Bekleme yönteminin doğru bir şekilde kontrol edilmesi için, yazmaç öbeği tasarımına her girdi için ek 1-bit geçerli biti ve etiket bilgisi eklenmiştir. Yazmaç Oku aşamasında

okuma yapılırken, okuma yapılacak yazmacın geçerli bitinin mantık-1 olması gerekmektedir. Buyruk için gerekli tüm yazmaçlar geçerli olmadığı sürece, Yazmaç Oku aşaması boru hattını durdurur ve yazmaçların geçerli olmasını bekler. Ayrıca, işlenen buyruk yazmaç öbeğine yazma işlemi gerçekleştirecek ise, ilgili yazmacın geçerli biti mantık-0 yapılır ve etiket bilgisi güncellenir. Geri Yaz aşamasında, yazma işlemi yapılacağı eğer yazmaç öbeğinin ilgili etiket bilgisi eşleşir ise ilgili geçerli biti mantık-1 yapılır. Etiket bilgisi, ilgili yazmacı hangi buyruğun en son geçersiz yaptığı bilgisini tutar. Bu sayede, hatalı geçerli yapma durumu önlenmiş olur. Etiket bilgisi olarak, buyruklara özel olan program sayacı ya da rastgele değerler kullanabilir. Tasarımda etiket bilgisi olarak, Yazmaç Oku aşamasında tutulan bir sayaç değeri kullanılmıştır. Boru hattının uzunluğu dikkate alındığında, veri bağımlılığı yaratabilecek buyruk sayısının 3 olduğu gözlemlenmiştir. Bundan dolayı etiket bilgilerinin çakışmaması için 2 bit uzunluğunda bir sayaç kullanılmıştır.

```
1: addi x1, x2, x3
2: addi x1, x4, x5
3: addi x6, x6, x1
```

Şekil 3.3: Veri Bağımlılığı Örneği

Yürüt. Yürüt aşaması, boru hattındaki buyrukların yürütüldüğü ve sonuç değerlerinin hesaplandığı aşamadır. Yürüt aşaması birçok alt birimden oluşur. Bu birimler Aritmetik Mantık Birimi (AMB), Tamsayı Çarpma Birimi - Tamsayı Bölme Birimi (TCB-TBB), Dallanma Birimi (DB), Bellek İşlem Birimi (BİB) ve X Birimidir (XB). Yürüt, mikroişleme göre ilgili işleçleri ilgili birime yönlendirir ve birim sonuç değerini hesaplayana kadar boru hattını durdurur. Hesaplanan sonuç değerini mikroişleme ekleyerek Geri Yaz aşamasına iletir. Ek olarak, dallanma öngörücüsünün yanlış öngörüde bulunduğu durumlar Yürüt aşamasında tespit edilir ve boru hattının doğru program sayıcından devam etmesi sağlanır. Bu birimlere ek olarak, Denetim Durum Yazmaç buyrukları da bu aşamada gerçeklenmektedir.

Aritmetik Mantık Birimi: AMB, basit mantık işlemlerini (XOR, AND, Kaydırma, vb.) tek çevrimde gerçekleştirir. Ek olarak, toplama ve çıkarma işlemleri de bu birimde gerçekleştirilir. Toplama ve çıkarma işlemi, ÖTR'de belirlendiği üzere Kogge-Stone (Kogge & Stone, 1973) algoritması ile 9 çevrimde gerçeklenmiştir. Kogge-Stone kullandığı transistör sayısı (NlogN) ve sahip olduğu max fanout değeri (N) sayesinde diğer toplayıcılara (Bent-Kung (Brent & Kung, 1982), Sklansky (Sankar & Ali, 2013), vb.) kıyasla tasarım isterlerine daha uygun olduğu belirlenmiştir.

Tamsayı Çarpma Birimi - Tamsayı Bölme Birimi: TCB ve TBB, çarpma, bölme ve kalan hesaplama buyruklarını sıralı bir şekilde hesaplamakla sorumludur. Çarpma işlemi 33, bölme işlemi ise 33 çevrimde hesaplanmaktadır.

Dallanma Birimi: Program akışını değiştiren buyrukların sonuçları elde edildiğinde, Getir2 aşamasındaki dallanma öngörücünün bu sonuç ile güncellenmesinden ve yanlış öngörü halinde Getir1 aşamasındaki program sayacının değerini güncellenip ve boru hattının boşaltılmasından sorumludur.

Bellek İşlem Birimi: BİB, Çöz aşamasında mikroişlemlere eklenen okuma ve yazma buyruk-

larının verilerinin maskelendiği ve buyruk tipine göre ayarlandığı birimdir. Bu birim buyruklar için gerekli okuma ve yazma isteklerinin bellek hiyerarşisine ve buyruk mimari kısıtlarına bağlı olarak oluşturulmasını sağlar.RV64 buyruk mimarisinde, bayt (*-ing byte*), yarım kelime (*-ing half word*), kelime (*-ing word*), çift kelime (*-ing double word*) olmak üzere 4 farklı bellek alan erişimi desteklenmektedir. Bu bellek erişimleri sırasıyla 8, 16, 32, 64 bitlik bellek okuması veya yazması anlamına gelir. Okuma veya yazma buyruğunun bellek alan erişimine bağlı olarak bellekten okuma veya yazma yapması için gerekli bellek istekleri bellek işlem birimi içerisinde belirlenir. Ayrıca okuma yapılan veriler için buyruğun işaretli genişleme ile okuması gerekli ise bununla ilgili bellek isteklerinin düzenlenmesi de bellek işlem birimi tarafından ayarlanır.

X Birimi: XB, yarışma şartnamesinde bulunan ek buyrukları gerçekleştirmekle görevlidir. Hamming uzaklığı hesaplama buyruğu (HMDST) yürütülürken her çevrim 4 bit kontrol edilerek toplamda 16 çevrimde sonuç hesaplanır. PKG, RVRS ve SLADD buyrukları tek çevrimde bit manipülasyon teknikleriyle hesaplanır. CNTZ ve CNTP buyrukları ise gelen işleçlere göre her çevrim 4 bit kontrol ederek farklı sayıda çevrimde hesaplanır.

Geri Yaz. Geri Yaz aşaması, boru hattının son aşamasıdır ve işlenen buyrukların sonuçlarının yazmaçlara yazıldığı aşamadır. Ek olarak, yazmaç öbeğinde yazma yapılan yazmacın etiket bilgisi ve buyruk bilgisi kontrol edilir ve eşleşme durumunda ilgili yazmacın geçerli biti 1 yapılır. Böylece, ilgili yazmacı okumak isteyen başka buyruklar okuyabilirler.

3.2.1.3. Denetim Durum Birimi. Denetim Durum Birimi (DDB) buyruğun işlenmesi esnasında yaşanabilecek olağan dışı durumları (ODD) kontrol etmekte, ilgili denetim durum yazmaçlarını güncellemekte ve boru hattını boşaltan sinyalleri belirlemektedir. (*-ing. pipeline flush*). Boru hattı aşamaları ile ilgili olarak iki ana olağan dışı durumdan söz edilebilir. İlk olarak, Çöz aşamasındaki buyruğun mevcut buyruk haritalamasına uygun olmaması ile buyruk geçersiz ise bu bir olağan dışı durum oluşturmaktadır. Olağan dışı bir durum oluştuğunda çekirdek mcause yazmacını ODD kodu ile ve mepc yazmacını ODD'ye yol açan buyruğun program sayacı ile günceller. Sonra trap vector (MTVEC) olarak adlandırılan, olağan dışı durumu çözecek kod parçasının başlangıç adresini içeren yazmacı okur ve bu adresten yeni buyrukları getirir. Buna ek olarak, ECALL ve EBREAK buyrukları kullanıcı programlarının yürütme ortamına çağrı yapması için kullanılır. Bu çağrılar tam anlamıyla bir olağan dışı durum ifade etmese de çağrılar olağan dışı duruma yol açacak şekilde gerçeklenmiştir.

Yürüt aşamasında denetim durum buyrukları denetim durum yazmaçlarını kullanmaktadır. DDY buyruklarındaki işlemler atomik olarak yapılmaktadır. Olası atlama durumunda nereye atlayacağı bilgisi vb. denetim durum yazmaçlarına kaydedilmektedir ve gerekli durumlarda boru hattı boşaltılmaktadır.

3.2.2. Bellek Tasarımları

Bellek hiyerarşisinde proje isterlerinde istenen 1.seviye buyruk önbelleği (L1B$), 1.seviye veri önbelleği (L1V$), L2 önbelleği (L2$) ve ana bellek için toplamda dört adet denetleyici (1.seviye buyruk önbelleği denetleyicisi (L1 BD), 1.seviye veri önbelleği denetleyicisi (L1 VD), L2 Denetleyici ve Ana Bellek Denetleyici) tasarlanmıştır. Sistem bellek işlemleri için denetleyicilerle haberleşmektedir. Bellek denetleyicileri ise kendi aralarında ve ilgili belleklerle haberleşme yap-

maktadır. Şekil 3.1 bellek hiyeraşisinin üst seviye blok şeması gösterilmektedir.

3.2.2.1. Önbellekler ve Arayüzleri. 1. seviye önbellekler için her satırı 128-bit olan 128 satırlı 2 KB bir önbellekte tam-ilişkili bir tasarım 128 karşılaştırıcı kullanırken, doğrudan-eşlemeli önbellek ise 1 karşılaştırıcıya sahip olacaktır. Öte yandan kümeli ilişki önbellek için doğrudan eşlemeliye göre gecikmesi daha az olsa da çekirdeğin yalınlık ilkesi göz önünde bulundurulduğunda daha karmaşık ve güç tüketimi daha yüksektir. Bu nedenle L1B$'nin ve L1V$'nin doğrudan eşlemeli olarak tasarlanması kararlaştırılmıştır. 2. seviye önbellek için 8 KB doğrudan eşlemeli önbellek kullanıldığında başarım kümeli ilişkili önbelleğe göre daha düşük olacaktır. İkinci seviye önbellekte aranan verinin bulunamaması durumunda veri isteği ana bellek hiyerarşisine gitmektedir. Bu durum işlemcinin performansını çok düşüreceğinden önbellek başarımı birinci seviye önbellek hiyerarşisine göre çok daha fazla önem taşımaktadır. Bu nedenle L2$'ın her bir yolu L1$ ile aynı iki boyutlu düzene sahip olacak şekilde (2KB boyutunda her satırı 128-bit olan 128 satırlı) 4-yollu kümeli-ilişkili sistemin kullanılması sistemin yalınlığını korumaya yardımcı olacaktır.

2. seviye önbellek ise 4-yollu kümeli ilişkili olarak tasarlanmıştır. 2. seviye önbelleklerdeki çıkarma politikası ise rastgele olarak gerçeklenmiştir. 1. seviye önbellekler ve dört yollu 2. seviye önbelleğin her bir yolu, veri öbeği (*-ing cache line*) 128-bit ve 128 satırlı 2KB olarak tasarlanmıştır. Önbelleklerin politikası geri yazma (*-ing write-back*) ve yaz ve yerini ayır (*-ing write-allocate*) olarak belirlenmiştir. Adres genişliği 32-bit olan sistemdeki önbellek haritalandırmasında, 128-bit (16 bayt) veri öbeğini indislemek için 4-bit, 128 satırı adreslemek için 8-bit ve ilgili adresin etiketi için de geriye kalan 21-bit kullanılmaktadır.

1. Seviye Önbellekler. 1. seviye önbelleklerin etiketleri için her satırı 21-bit genişliğinde olan 128 satırlı ve 2.seviye önbelleklerin etiketi için ise her satırı 84-bit genişliğinde olan 128 satırlı etiket önbellekleri ayrı olarak tasarlanmıştır. Oluşturulan önbelleklerin daha detaylı bilgilerine Bölüm 4.2'ten ulaşılabilir.

1. seviye buyruk önbelleği denetleyicisi Getir1, Getir2 ve buyruk önbelleği ile, 1. seviye veri önbelleği denetleyicisi ise adres tekleyici birimi ile haberleşmektedir. Ek olarak, 1. seviye önbellek denetleyicileri kendi önbelleklerinde istenen veriyi bulamadıkları durumda 2.seviye önbellek denetleyicisiyle haberleşmektedir. 1.seviye önbelleklerin sakla (*-ing store*) ve yükle (*-ing load*) işlemlerindeki üst seviye davranışı aşağıda maddeler olarak açıklanmıştır.

(i) 1.seviye önbellek denetleyicileri, gelen isteğin önce önbellekte olup olmadığını sorgulamak için veri önbelleklerine ve etiket önbelleklerine adresteki satır için ayrılan kısımla önbelleklere sorgu yapar.

(ii) Bir sonraki çevrimde sorgu sonucu gelen etiket ile istek yapılan adresin eşitliği karşılaştırılır. Eğer ilgili adres önbellekte bulunduysa ve satır geçerli ise;

  (a) Yazma isteği yapıldıysa giriş olarak alınan maskelemeye göre 128-bitlik verinin ilgili bölümüne yeni veri yazılır ve kirli bit (*-ing dirty-bit*) mantık-1 yapılır.

  (b) Okuma isteği yapıldıysa istek adresinin veri öbeği kısmından itibaren veri için 64, buyruk için 32-bit ilgili birime yollanır. Aynı çevrimde 1.seviye önbelleğe tekrar okuma isteği yapıldıysa, denetleyici bu isteği kabul eder ve diğer çevrimde

sorgu sonucunda yeni istek önbelleklerde bulunuyorsa çevrim kaybetmeden veriyi çekirdeğe yollamaktadır.

Eğer ilgili adres önbellekte bulunmadıysa 2.seviye önbelleğe okuma isteği gönderilir, veri geldikten sonra 1.seviye önbelleğin ilgili satırına 2.seviye önbellekten gelen veri ve etiket yazılır. Yazılan yerdeki veri eğer geçerli ve kirli ise ikinci seviye önbelleğe bu veri geri yazılır, değilse işlem yapılmaz. Çekirdekten gelen isteğe göre madde 2a veya 2b adımı gerçekleştirilir.

2.Seviye Önbellek. 2.seviye önbellek denetleyicisi; 1.seviye önbellek denetleyicileri ve ana bellek denetleyicisi ile haberleşmektedir. 2.seviye önbelleklerin çıkarma politikası rastgele olarak belirlenmiştir. Buradaki rastgeleliği sağlamak için 2-bitlik her çevrim bir değer artan sayaç kullanılmıştır. Sayacın değerine göre çıkarılacak satırın olu belirlenir. 2.seviye önbellek denetleyicisi, 1.seviye önbellek denetleyicisinden istek alarak işleme başlamaktadır. Eğer aynı anda iki 1.seviye önbellek denetleyicisinden de istek yapıldıysa öncelik olarak buyruk önbellek denetleyicisinin isteği alınır ve o anki veri önbellek denetleyicisinin isteği kaydedilir. Buyruk denetleyicisinin isteği yürütüldükten sonra kaydedilen veri denetleyicisinin isteği gerçekleştirilir. Aşağıda maddeler olarak 1.seviye önbellekten istek yapıldığında gerçekleştirilen aksiyonlar listelenmiştir.

(i) Gelen isteğin önce önbellekte olup olmadığını sorgulamak için tüm yollara ve etiket önbelleğine adresteki satır için ayrılan kısımla sorgu yapılır.

(ii) Bir sonraki çevrimde sorgu sonucu gelen etiket ile istek yapılan adresin etiketi her yol için karşılaştırılır. Eğer ilgili adres önbellekte bulunduysa ve satır geçerli ise;
   (a) Yazma isteği yapıldıysa bulunan yoldaki 128-bitlik verinin üzerine yeni veri yazılır ve kirli bit (*-ing dirty-bit*) mantık-1 yapılır.
   (b) Okuma isteği yapıldıysa bulunan yoldaki 128-bitlik veri 1.seviye önbelleğe gönderilir.

   Eğer ilgili adres herahngi bir yolda bulunmadıysa, ana bellek denetleyicisine okuma isteği yapılır. Veri geldikten sonra ilgili satır için;
   (a) Herhangi bir yolda geçersiz veri var ise oraya ana bellek denetleyiciden gelen veri yazılır ve aynı çevrimde 1.seviye önbelleğe yazma isteği gönderilir.
   (b) Geçersiz yol yok ise, rastgele bir yol seçilir ve oraya ilgili veri yazılıp, 1.seviye önbelleğe yazma isteği gönderilir. Eğer çıkarılan yoldaki veri kirli ise aynı çevrimde ana bellek denetleyicisine de yazma isteği yapılır.

Tablo 3.1'de önbelleklerin yapılan istekleri en düşük ve en yüksek gecikmeleri çevrim cinsinden gerçekleştirme süreleri verilmiştir. Çekirdek bir bellek isteği yaptığında eğer veri 1.seviye önbelleklerde bulunuyorsa ilk istek iki çevrimde gönderilir. Ardışık istekler yapıldığında ise performans kaybını azaltmak için önceki okuma isteği tamamlandığında yeni okuma isteği o çevrim kabul edilmektedir. Fakat yazma istekleri için her zaman gecikme iki çevrimdir. 1.seviye önbelleklerde veri bulunamadığında bellek isteği 2.seviye önbellek isteğine dönüşür. Bu durumda 2.seviye önbellekte veri bulunursa toplamda 6 çevrimde, eğer bulunamazsa ana belleğe gidileceğinden 14 çevrimde istek gerçekleştirilir.

3.2.2.2. Ana Bellek Arayüzü. Ana bellek denetleyicisi, 2.seviye önbellek denetleyicisi ve ana bellek ile haberleşmektedir. Ana bellek denetleyicisi, bir üst hiyerarşi olan 2.seviye önbellek denetleyicisinden gelen istek ile işlem yapmaya başlamaktadır. 2.seviye önbellek ile arasında

Tablo 3.1: Önbelleklerden Verilerin Çekilme Süresi

Önbellek Türü	En Düşük Gecikme (Çevrim)	En Yüksek Gecikme (Çevrim)
1.seviye önbellek	1	2
2.seviye önbellek	6	14

128-bitlik veri ile 32-bitlik adres sinyalleri bulunmaktadır. Ana bellek denetleyicisi ile ana bellek arasında 32-bitlik veri ve 32-bitlik adres sinyalleri yer almaktadır.

Ana bellek denetleyicisi, 2.seviye önbellekten gelen 128-bitlik veriyi önce dört adet 32-bitlik veriye böler, daha sonrasında her bir 32-bitlik veriyi şifrelemek için aynı anda dört adet SBOX kullanarak şifreler ve aynı çevrimde şifrelenmiş veriyi ana belleğe yazar. Bu işlem dört kez tekrarlanarak ana belleğe yazma işlemi tamamlanır. Eğer 2.seviye önbellek denetleyicisi, ana bellekten veri isterse, ana bellek denetleyicisi 32-bitlik şifreli veriyi ana bellekten alır ve dört adet ters SBOX birimi ile şifreyi çözerek içerisindeki ara belleğe saklar. 128-bitlik veri elde edildiğinde ise 2.seviye önbelleğe veriyi gönderir.

SBox. SBox modülü 8 bitlik bir girişi, 8 bitlik bir çıkışa eşler. Tasarımda bir tablolardan 32 bite karşılık gelecek şekilde her sekiz biti için bir adet sbox modülü çağrılmıştır. SBox tablosu isimli modülün içine tablo gömülmüştür ve ana modülden 4 kere çağrılmıştır. Ters SBox ise şifrelenmiş 8 bit veriyi çözmek için kullanılmaktadır. İşlemler ile hesaplama sonucunda dinamik şekilde çeviri işlemini yapan bir devrenin gecikmesi tablo ile statik değerlerle gelen veriyi işleme sokarak gerçeklenen tasarıma kıyasla daha yüksek gecikmeye neden olur. Bunun sebebi yapılacak hesaplamaların devre düzeyinde karmaşıklığı artırmasıdır. Buna karşılık tablo tabanlı SBox biriminin kullandığı alan, diğer tasarıma kıyasla daha fazladır. Tasarım genelinde performans artışının daha önemli olduğu göz önünde bulundurularak tablo tabanlı birimin gerçeklenmesine karar verilmiştir.

3.2.3. Çevre Birimleri Tasarımları

Tasarlanan işlemci, önbellek ve ana belleklere ek olarak, giriş çıkış birimlerine bağlıdır. İşlemci, bu giriş çıkış birimleri üzerinden ilgili protokolleri kullanarak işlemci dışı ile iletişime geçebilir. Şekil 3.1'de görüldüğü gibi işlemci çekirdeği Adres Tekleyici birimi üzerinden UART ve SPI denetleyicilerine bağlıdır. Bu sayede, programcı gerekli buyrukları kullanarak bu iletişim protokolleri ile çeşitli cihazlarla iletişime geçebilir.

3.2.3.1. Adres Tekleyici. Adres tekleyici (AT) birimi, boru hattı ile çekirdeğin dışı arasındaki bağlantıyı sağlamaktadır. Boru hattının Yürüt aşamasında bulunan Bellek İşlem Birimi (BİB), işlediği yükle/sakla (LD/SD) buyruklarını AT birimine iletir. AT birimi yükle/sakla isteklerinin hedef adreslerine göre ilgili çekirdek dışı birimlere yönlendirir. Bu birimler 1. seviye veri önbelleği, UART ve SPI denetleyicileridir. AT biriminin moduler tasarımı sayesinde ek çekirdek birimleri (giriş çıkış birimleri, hızlandırıcılar vb.) eklenebilmektedir. Ek olarak, AT birimi gerekli durumlarda BİB'i bekleterek boru hattının duraklamasını sağlayabilir. Bu bariyer özelliği çekirdek dışı birimlerin senkronizasyonunu sağlar ve çalıştırılan programın doğruluğu için gereklidir.

3.2.3.2. UART. UART kendi içeiresinde 3 farklı alt görev birimi içerir. Bunlar UART denetleyici, UART verici ve UART alıcıdır. Şartnamede belirtildiği üzere UART tasarımını yaparken 32x8 bitlik değerleri saklayabilecek alıcı ve verici tarafı için ayrı şekilde 2 farklı ilk giren ilk çıkar yöntemi ile çalışan ara bellek (*-ing buffer*) kullanılmıştır. Bu arabellekler ve UART için gerekli olan kontrol yazmaçları UART denetleyici üzerinde uygulanmıştır. Denetleyici ara belleklerin durumuna ve adres tetikleyiciden gelen yazma veya okuma isteklerine bakarak UART'ın temel birimleri olan alıcı ve verici birimlerin girişlerini sürer. Gerekli alma veya verme görevi UART denetleyici tarafından tespit edildiğinde UART alıcı ve verici birimleri çalışır ve çıktılara bağlı olarak UART denetleyici içerisinde tutulan UART yazmaçları ve ara bellekleri güncellenir. UART alıcı ve verici birimleri 1 adet başlangıç biti ve 1 adet bitiş biti olan standart UART protokolüne göre çalışmaktadır.

3.2.3.3. SPI. SPI Denetleyici içerisinde 5 farklı yazmaç bulundurur. Bu yazmaçlar SPI arayüzünün farklı konfigürasyonlarını kontrol etmek ve iletim arabelleklerinde değer saklamak için kullanılır. SPI denetleyicinin kontrol sinyalleri Adres Tekleyici ile denetlenir. İşlemciden gelen seri yazma istekleri SPI Veri Yazma Yazmacında depolanır. SPI Komut Yazmacının içerdiği direction, "01" değerini aldığında seri iletim başlar. Seri okuma işlemi için direction "10" değerini alır ve okunan değer Veri Okuma Yazmacına kaydedilir. İşlemci Yazmaç okuma isteği ile bu yazmaçı paralel şekilde okuyabilmektedir.

4. ÇİP TASARIM AKIŞI

Proje isterleri gereği, tasarlanan işlemcinin SKY130 teknolojisine uygun serimi gerçekleştirilmelidir. Serim işlemi için SKY130 teknolojisini destekleyen açık kaynaklı OpenLane akışı (Ghazy & Shalan, 2020) tercih edilmiştir. OpenLane akışı, birçok açık kaynaklı aracı kullanarak yazmaç seviyesinde yazılan modülleri istenilen üretim teknolojisinde ve köşesinde serime uygun hale getirmektedir. Proje kapsamında, OpenLane aracı, tasarlanan işlemciyi SKY130 teknolojisinde ve TT (Typical - Typical) - 25°C - 1.8V köşesinde serime uygun hale getirilmiştir.

İşlemcinin serim işlemini gerçekleştirmek için OpenLane akışının sunduğu makrolama yöntemi (OpenLane, 2022b) tercih edilmiştir. Bu yöntemde, her bir birim ayrı bir şekilde makrolaştırılır. İşlemcinin birimler ayrı bir şekilde OpenLane akışından geçirilip makrolaştırılmıştır. Bu makrolara ek olarak, önbellek hiyerarşisi için bellek makroları gerekmektedir. Bellek makrosu olarak SKY130 teknolojisine ait yazmaç dizilerini *(-ing, D-Flipflop Array)* kullanmak alan ve enerji açısında verimli olmayacağından dolayı, SKY130 SRAM teknolojisi tercih edilmiştir. SRAM makrolarını oluşturmak için SKY130 teknolojisini destekleyen açık kaynaklı OpenRAM aracı (Guthaus et al., 2016) kullanılmıştır. Son olarak, oluşturulan birim ve SRAM makrolarının, ana bir modül altında toplanarak, OpenLane akışı ile serime uygun hale getirilmiştir (OpenLane, 2022a).

4.1. Makro Tasarım Akışı

İşlemciyi serime uygun hale getirmek için, ilk olarak işlemcinin birimlerinin makrolaştırılması gerekmektedir. Proje isterleri gereği, oluşturulan makroların SKY130

kurallarına (DRC, maksimum fanout, maksimum cap, vb.) uygun olması gerekmektedir. Makrolaştırma işleminde en fazla karşılaşılan hata maksimum fanout hatasıdır. Bu hata, tek bir kapıdan çıkan bir sinyalin belli bir sayıdan fazla kapıda giriş olarak kullanılmasından dolayı oluşur. Bu hatayı çözmek için ise serimdeki sinyallerin uzunlukları sınırlandırılmıştır. Böylece, herhangi bir çıkış sinyali belli bir sayıda kapıya ulaşamadan arabellekler *(-ing, buffer)* kullanılarak dağıtılır. Ancak, koyulan arabellekler makronun kritik yol uzunluğunu ve alanını arttırabilir. Makrolaştırma işleminde karşılaşılan ikinci bir hata ise anten hatasıdır. Bu hata OpenLane tarafından anten diyotları eklenerek çözülebilmektedir. Fakat, bazı durumlarda alanın yetersiz ya da hedef yoğunluğun fazla olmasından dolayı tüm anten hataları çözülememektedir. Bu durumlarda makro alanı arttırılarak ya da hedef yoğunluğu azaltılarak, anten hataları çözülmüştür.

İşlemcinin alt birimleri ayrı olarak incelenmiş ve hataları çözülmeye çalışırken kritik yol uzunluğu ve alan kullanımı azaltılmaya çalışılmıştır. 2. seviye önbellek denetleyici ve çekirdek birimleri dışındaki tüm birimler hatasız bir şekilde akıştan geçirilmiştir. Projenin devamında 2. seviye önbellek denetleyicisi ve çekirdek birimlerinin hataları üzerine çalışılmaya devam edecektir. Ek olarak, hatasız makrolaştırılan birimlerin alan kullanımı, güç tüketimi ve kritik yol uzunlukları optimize edilmeye çalışılacaktır. İşlemcinin birimlerinin makro sonuçları tablo 4.1'de verilmiştir.

Tablo 4.1: İşlemci Birim Makrolarının Sonuçları

Birim	Boyut(um - um)	Kritik Yol(ns)	Güç(mW)
Çekirdek	5387 - 3247	15.06	140
L1B Denetleyici	1252 - 1262	62.93	26.1
L1V Denetleyici	1391 - 1401	73.14	33.4
L2 Denetleyici	3914 - 1387	12.41	52.4
Ana Bellek Denetleyici	1050 - 1060	29.35	14.1
Adres Tekleyicisi	328 - 339	5.73	3.24
UART Denetleyici	566 - 576	8.78	0.55
SPI Denetleyici	701 - 711	16.30	12

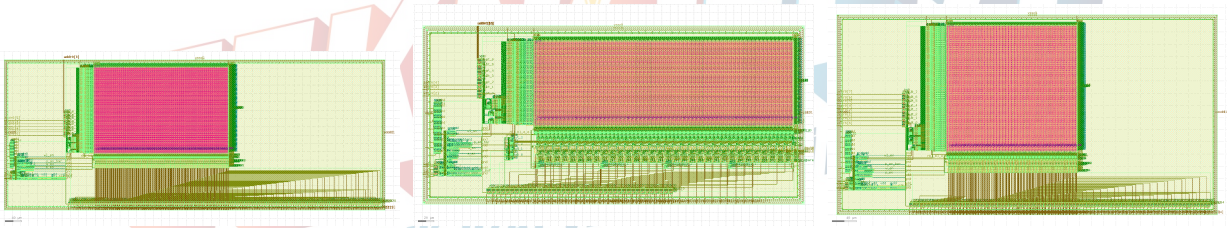
4.2. SRAM Tasarım Akışı

İşlemci seriminde kullanılacak SRAM makrolarının SKY130 teknolojine uygun bir şekilde hazırlanması için açık kaynaklı OpenRAM araçı kullanılmıştır. 3.2.2 Bellek Tasarımları bölümünde açıklandığı üzere, işlemci tasarımında 2 adet her satırı 128-bit olan 128 satırlı 2 KB boyutunda 1. seviye önbellek ve 1 adet 4 yollu her yol satırı 128-bit olan 128 satırlı 8 KB boyutunda 2. seviye önbellek kullanılmıştır. 1./2. seviye önbelleklerde tutulan veri ve buyrukların etiket bilgilerini saklamak için, her satırı 21/84-bit olan 128 satırlı ek bellekler kullanılmıştır. Kullanılan bellek birimleri için hem makro alanını hem de giriş/çıkış sayısını azaltmak amacıyla 1 adet oku/yaz bağlantısı (1rw) olan bellek birimleri tercih edilmiştir. Açıklanan önbellek hiyerarşisi için 3 farklı boyutta SRAM bellek modülüne ihtiyaç duyulmaktadır.

- **L12 Önbelleği** her satırı 128-bit olan 128 satırlı 2 KB boyutunda SRAM dizisinden oluşur. Bu boyuttaki önbellek OpenRAM kütüphanesinde hazır olarak bulunan bir makro olmadığı

için özel olarak akıştan geçirilmesi gerekmektedir. Ancak, OpenRAM aracındaki bazı hatalardan dolayı, her satırı 128-bit olan 128 satırlı bir SRAM dizisi oluşturulamamıştır ve 129-bit olan 129 satırlı SRAM dizisi oluşturulmuştur. Ek bir satırın ve sütunun alan olarak önemli bir fark yaratmamasına rağmen, modulün adres girişini, veri okuma girişini ve veri yazma çıkışını birer bit arttırmıştır. Ortaya çıkan bu sorun basit bir şekilde, gereksiz girişlerin mantık-0'a bağlanmasıyla çözülmüştür. Oluşan SRAM biriminin serimi şekil 4.1a'de görülmektedir.

- **L1 Etiket Belleği** her satırı 21-bit olan 128 satırlı SRAM dizisinden oluşur. Bu boyuttaki önbellek OpenRAM aracında hazır olarak bulunan bir makro olmadığı için özel olarak akıştan geçirilmesi gerekmektedir. Ancak, OpenRAM aracındaki bazı hatalardan dolayı, her satırı 21-bit olan 128 satırlı bir SRAM dizisi oluşturulamamıştır ve 75-bit olan 129 satırlı SRAM dizisi oluşturulmuştur. Oluşan SRAM makrosu incelendiğinde, gereksiz SRAM sütunlarının alana önemli derecede arttırdığı görülmüştür ve OpenRAM kütüphanesinde hazır olarak bulunan her satırı 32-bit olan 256 satırlı SRAM makrosunun daha az kapladığı tespit edilmiştir. Bu sebepten dolayı, tasarımda OpenRAM kütüphanesinde hazır olarak bulunan her satırı 32-bit olan 256 satırlı SRAM makrosu kullanılmıştır. Gereksiz girişler, modülün işlevselliğini bozmaması için mantık-0'a bağlanmıştır. Kullanılan SRAM biriminin serimi şekil 4.1b'de görülmektedir.
- **L2 Etiket Belleği** her satırı 84-bit olan 128 satırlı SRAM dizisinden oluşur. Bu makroyu oluştururken, L12 Önbelleğinde karşılaşılan sorunlar karşılaşılmıştır ve benzer şekilde çözülmüştür. Sonuç olarak, her satırı 85-bit olan 129 satırlı bir SRAM makrosu oluşturulmuştur ve gereksiz girişleri mantık-0'a bağlanmıştır. Oluşturulan SRAM biriminin serimi şekil 4.1c'de görülmektedir.



(a) L12 Önbelleği (b) L1 Etiket Belleği (c) L2 Etiket Belleği

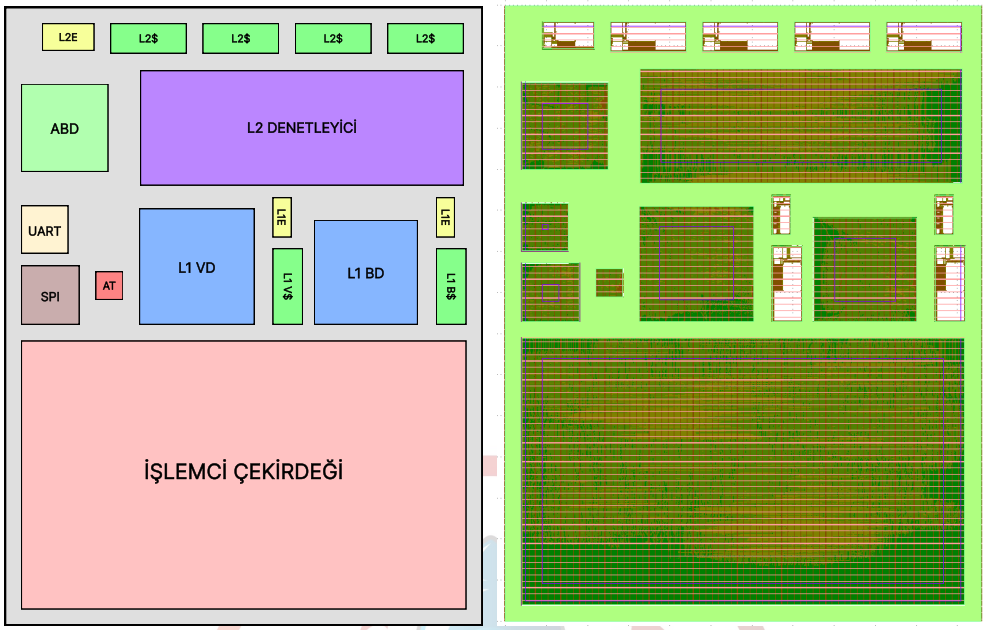
Şekil 4.1: SRAM Makroları

Yukarıda açıklanan SRAM makroları, önbellek denetleyicilerine uygun şekilde bağlanarak önbellek hiyerarşisini oluşturmaktadır. 1. seviye veri/buyruk önbelleği, veri/buyruk bilgilerini bir adet L12 önbelleğinde, etiket bilgilerini bir adet L1 etiket belleğinde tutar. 2. seviye önbellek, veri bilgisini 4 adet L12 önbelleğinde, etiket bilgisini L2 etiket belleğinde tutar.

4.3. İşlemci Tasarım Akışı

İşlemci serimi oluşturma işleminin son aşaması olarak, oluşturulan birim ve SRAM makroları tek bir ana modül altında toplanmıştır. Ana modülde gerekli bağlantılar yapılıp, OpenLane akışına makroların gerekli girdileri (konum, saat ağı, GDS dosyası, LEF dosyası, vb.) verilmiştir. Şekil 4.2'de makroların konumları (sol) ve oluşan serimi (sağ) gösterilmiştir (Görsel

karmaşıklığından dolayı sadece MET4 ve MET5 gösterilmiştir.).



(a) (b)

Şekil 4.2: İşlemci Serimi

5. TEST

Proje kapsamında tasarlanan işlemcinin işlevselliği üç farklı aşama ile doğrulanmaktadır: (i) RTL seviyesindeki simulasyon, (ii) FPGA'de gerçekleme ve (iii) serim sonrası kapı seviyesi *(-ing gate-level* simulasyon. Her bir aşamada yarışma komitesi tarafından paylaşılan üst modül (wrapper) ana modül olarak kullanılması planlanmıştır. X buyruklarını test edebilmek için RISC-V derleyicisine X buyrukları eklenmiş ve oradan üretilen testler ile X buyruklarının işlevselliği doğrulanmıştır. Karıştırma testleri için ise test programlarının buyrukları SBOX algoritması ile karıştırılarak yeniden düzenlenmiş ve tüm test aşamaları bu şekilde gerçekleştirilmiştir.

5.1. RTL Seviyesindeki Simulasyon Ortamı

RTL seviyesindeki simulasyon ortamının diğer ortamlara göre çok daha ayrıntılı bir şekilde hata ayıklama gerçekleştirilebildiğinden ilk aşama olarak belirlenmiştir. Simulasyon ortamı olarak AMD-Xilinx şirketinin Vivado aracı kullanılmıştır.

Bu ortamda her bir boru hattı aşaması ve bellek hiyerarşisinin her bir parçasının üst-düzey işlevselliği test edilerek işlemcide bağlanmıştır. Bu adımdaki amaç, test durumlarına göre istenilen çıktıların doğruluğunu göstermek ve modüller arası entegrasyonu sağlayarak işlemciyi çalıştırmaktır. Öncelikli olarak dikkate alınan işlevsellikler: (i) Getir-1, 1.seviye buyruk önbelleği denetleyicisi ve Getir-2 arasındaki program sayacı güncelleme ve buyruğun doğru aktarımı, (ii) Getir-2'de bulunan dallanma öngörücüsünün işlevselliği ve sıkıştırılmış buyrukları doğru bir şekilde yönetebilmesi, (iii) Çöz aşamasının buyrukları doğru bir şekilde çözebilmesi, (iv) yazmaç oku, yürüt ve geri yaz aşamalarının gelen buyruklar arasında veri bağımlılığı olduğu durumlarda

boru hattını duraklatıp hatasız ilerlemesi ve (v) yürüt içindeki birimlerin hesaplamaları doğru bir şekilde yapabilmesi test edilmiştir.

İşlemcinin parçaları bağımsız bir şekilde test edildikten sonra bu aşamadaki bir sonraki adım ise boru hattını, bellek denetleyicilerini ve bellek birimlerini bağlayıp işlemcinin parçalarının aralarındaki iletişiminin *(-ing handshaking* doğruluğunu test etmektir. Bu adımda eksiksiz bir doğrulama için RISC-V tarafından sağlanan her bir buyruk ve buyruk mimarisi için (köşe durumları dahil *(-ing corner case*) açık kaynak olarak sağladığı testler kullanılmıştır. RISC-V tarafından sağlanan RV64IMC buyruk kümesi için bulunan tüm testler (toplamda 67 adet) gerçekleştirilmiştir. RISC-V'ın dokümantasyonunda da belirtildiği üzere, tasarlanan işlemci her bir testte bulunan birçok test durumlarını yürütmüş ve istenilen yazmaç öbeği çıktısına program sonunda erişmiştir ($x17->93, x10->0$).

5.2. FPGA Test Ortamı

Tasarlanan işlemci RTL seviyesindeki simulasyonları başarıyla geçtikten sonra iki farklı FPGA ortamında (ZedBoard (Xilinx, 2019b) ve VCU108 (Xilinx, 2019a)) test edilmiştir. İşlemci iki FPGA kartında da 125MHz saat sıklığında gerçeklenmiştir. Bu testi gerçekleştirebilmek için öncelikle UART üzerinden test edilecek buyruklar ana belleğe aktarılmış ve oradan veriler çekilmiştir. Testlerin başarıyla geçtiğini anlamak için ilgili yazmaçlar ledler üzerinden sürülüp testin doğruluğu incelenmiştir. RTL simulasyonunda olduğu gibi tasarlanan işlemci tüm RISC-V testlerini başarıyla FPGA ortamında da geçmiştir.

5.3. Serim Sonrası Kapı Seviyesi Test Ortamı

FPGA ortamında yapılan testlerin başarıyla sonuçlanmasından sonra serim sonrası testlerin yapılmasına başlanmıştır. Fakat Bölüm 4'te de bahsedildiği üzere stabil bir serim elde edilemediğinden bu aşama daha gerçekleştirilememiştir.

6. TAKIM ORGANİZASYONU

6.1. Takım Organizasyonu

Proje takımı dört kişiden ve danışmandan oluşmaktadır. Proje danışmanı TOBB Ekonomi ve Teknoloji Üniversitesi Bilgisayar Mühendisliği bölüm başkanı olan Prof. Dr. Oğuz Ergin'dir. Proje ekip üyeleri Fatma Nisa Bostancı, İsmail Emir Yüksel, Yahya Can Tuğrul ve Alperen Bolat TOBB Ekonomi ve Teknoloji Üniversitesi Bilgisayar Mühendisliği bölümünde 2. sınıf tezli yüksek lisans öğrencileridir. Ekip üyeleri bilgisayar mimarisi, bellek teknolojileri ve donanım güvenliği alanlarında uluslararası hakemli konfersanlarda/dergilerde çıkmış birçok yayına sahiptir. Proje ekibi TOBB Ekonomi ve Teknoloji Üniversitesinde bulunan Kasırga Mikroişlemciler Laboratuvarında bilgisayar mimarisi ve organizasyonu üzerinde araştırmalara ve çalışmalara devam etmektedirler.

6.2. Görev Dağılımı

Proje görev dağılımı genel hatlarıyla ÖTR'de belirtilen şekilde gerçekleşmiştir. Ancak belirtilen iş paketlerini birden çok kişi aynı anda gerçekleştirmiştir. İşlemci tasarımı ve verilecek kararlar takım olarak konuşulmuş ve kararlaştırılmıştır. Verilog HDL ile sayısal tasarımını

kısmı kendi içinde birkaç alt gruba ayrılmıştır. Boru hattı aşamalarını takım içinde alt birimlere ayrılarak kodlaması yapılmıştır ve herkes katılım göstermiştir. Bunun dışında bellek hiyerarşisi ve tasarımını İsmail Emir Yüksel ve Yahya Cam Tuğrul, çevre birimlerini ise Alperen Bolat ve Fatma Nisa Bostancı Verilog HDL ile gerçekleştirmiştir. FPGA üzerinde sınama testlerini ve doğrulamasını Alperen Bolat ve İsmail Emir Yüksel, programlama arayüzü ile test ortamının ve RISC-V sınama programlarının oluşturulması ve gerçekleştirilmesini Fatma Nisa Bostancı ve son olarak OpenLane aracı ile tasarımın silikon basım testlerinin ve aşamalarının gerçekleştirilmesini Yahya Can Tuğrul yapmıştır. Ekip karşılaştığı sorunları birlikte çalışarak çözmüş ve başarılı bir şekilde çalışan RV64ICMX işlemcisi geliştirilmiştir.

7. İŞ PLANI ve RİSK PLANLAMASI

Tablo 7.1: İş Planı Tablosu.

Görev	Bitiş Tarihi
İş Paketi 1'i tamamlamak	07.04.2022
Ön tasarım raporunu teslim etmek	11.04.2022
İş Paketi 2'yi tamamlamak	17.06.2022
İş Paketi 3'ü tamamlamak	24.06.2022
İş Paketi 4'ü tamamlamak	29.06.2022
İş Paketi 5'i tamamlamak	07.07.2022
Detay Teslim Raporunu Teslim Etmek	10.07.2022

Proje kapsamında iş paketlerinin, hedeflenen başlangıç tarihleri ve bitiş tarihleri Ön Tasarım Raporunda verilmiştir. Yine ÖTR'de belirtilen iş paketlerinin bitiş tarihlerini Tablo 7.1'de bulabilirsiniz. Ertelenen tarihler de göz önünde bulundurularak ÖTR'de belirtilen planlanan bitiş tarihleri ve gerçek bitiş tarihleri arasında farklılıklar olmuştur. Yeni tarihlere göre hedeflenen tarihler için yeni bir düzenleme yapılmıştır.

İş Paketi 1, yonganın fikirsel tasarımını kapsamaktadır. Yonga tasarımı yarışma isterlerine göre belirlenmiş ve fikirsel sistem tasarımı yapılmıştır. Boru hattı ve yonga çevre birimleri birlikte teorik olarak tasarlanmış, sistem gereksinimleri belirlenmiştir. İş paketi 1 tam olarak ve hedeflenen tarihten önce bitmiş ve buna bağlı olarak ÖTR teslim edilmiştir.

İş Paketi 2, HDL tasarımın gerçekleştirilmesini kapsamaktadır. Ara rapor ile birlikte belirlenen tasarım kararları detaylandırılıp, donanıma özel modüllerin giriş/çıkış'ları ve mimarideki görevleri ayrıntılı tasarlanmıştır. Oluşturulan modüller, planlanan tasarıma uygun bir şekilde Verilog HDL ile kodlanmıştır. İş Paketi 2 kendinden sonra gelen diğer paketleri çoğunlukla bağlamaktadır ancak kısmen İş Paketi 3 ve 4 ile iç içe gerçekleştirilebilmiştir. İş Paketi 2 büyük çoğunlukla bitmiştir. Yapılan testlere bağlı olarak son teslim tarihine kadar ufak tefek değişiklikler yapılabilmektedir. Ancak ana hatlarıyla bitmiştir. Bitiş tarihi ise hedeflenen bitiş tarihinden daha geç bitmiştir.

İş Paketi 3 Simülasyon seviyesinde doğrulamayı, donanım birimleri için geliştirilen HDL tasarım ve tüm çevre birim blokları birleştirmeyi, tüm sistemin doğrulaması için simülasyon

testleri gerçekleştirmeyi kapsamaktadır. Bu iş paketi güncellenen hedeflenen bitiş tarihinden önce bitmiştir ve tespit edilen hatalar büyük çoğunlukla düzeltilmiştir. İş paketinin tamamı bitmiştir.

İş Paketi 4, yonganın FPGA üzerinde demo edilmesini kapsamaktadır. Doğrulanan yonga tasarımı FPGA (Zedboard ve VCU108) üzerinde denenmiş ve demosu gerçekleştirilmiştir. FPGA üzerinde RISC-V testlerinden beklenen çıktıların elde edilip edilemediği incelenmiş ve tasarımW doğrulanmıştır. Bu iş paketi güncellenen hedeflenen bitiş tarihinden önce bitmiştir. İş paketinin tamamı bitmiştir.

İş Paketi 5, Yonganın GDSII dosyasının oluşturulmasını kapsamaktadır. Doğrulanmış ve sentezlenmiş yonga tasarımların tanımlamaya uygun olduğu Openlane (ve diğer sentez yardımcıları ile) ile onaylanmış ve üretim için son aşama olan GDSII dosyası elde edilmiştir. Bu iş paketi hedeflenen bitiş tarihinden önce biraz daha geç bitmiştir. İş paketinin tamamı bitmiştir.

KAYNAKÇA

Booth, A. D. (1951). A signed binary multiplication technique. *Quarterly Journal of Mechanics and Applied Mathematics*, *4*, 236-240.

Brent, & Kung. (1982). A regular layout for parallel adders. *IEEE Transactions on Computers*, *C-31*(3), 260-264. doi: 10.1109/TC.1982.1675982

Gao, H., & Zhou, H. (2005). Adaptive information processing: An effective way to improve perceptron branch predictors. *Journal of Instruction-Level Parallelism*, *7*, 1–10.

Ghazy, A., & Shalan, M. (2020). Openlane: The open-source digital asic implementation flow. In *Proc. workshop on open-source eda technol.(woset)*.

Guthaus, M. R., Stine, J. E., Ataei, S., Chen, B., Wu, B., & Sarwar, M. (2016). Openram: An open-source memory compiler. In *2016 ieee/acm international conference on computer-aided design (iccad)* (p. 1-6). doi: 10.1145/2966986.2980098

Kogge, P. M., & Stone, H. S. (1973). A parallel algorithm for the efficient solution of a general class of recurrence equations. *IEEE Transactions on Computers*, *C-22*(8), 786-793. doi: 10.1109/TC.1973.5009159

OpenLane. (2022a). *Chip Integration*. `https://openlane-docs.readthedocs.io/en/rtd-develop/doc/chip_integration.html`.

OpenLane. (2022b). *Hardening Macros*. `https://openlane-docs.readthedocs.io/en/rtd-develop/doc/hardening_macros.html`.

RISC-V. (2010). *RISC-V ISA Tests*. `https://github.com/riscv-software-src/riscv-tests`.

Sankar, D. R., & Ali, S. A. (2013). Design of wallace tree multiplier by sklansky adder. *International Journal of Engineering Research and Applications (IJERA)*, *3*(1), 1036–1040.

Takagi, N., Kadowaki, S., & Takagi, K. (2005). A hardware algorithm for integer division. In *17th ieee symposium on computer arithmetic (arith'05)* (pp. 140–146).

Xilinx. (2019a). *VCU108 Evaluation Board*. `https://www.xilinx.com/support/documents/boards_and_kits/vcu108/ug1066-vcu108-eval-bd.pdf`.

Xilinx. (2019b). *Xilinx ZedBoard Zynq-7000*. `https://www.xilinx.com/products/boards-and-kits/1-8dyf-11.html`.