**Abstract**

Bu laboratuvar, Vitis™ Yüksek Düzeyli Sentez (HLS) araç tasarım akışıyla ilgili en yaygın süreçlerin nasıl gerçekleştirileceğini açıklar. Bu laboratuvar, laboratuvarı tamamlamak için gerekli olan ZCU104 ve VCK190 kartlarını hedefler.

Bu laboratuvar yaklaşık 60 dakika sürmelidir.

**Yalnızca CloudShare Kullanıcıları**

Bir laboratuvara erişmeniz için size üç deneme hakkı verilir ve her bir laboratuvarı tamamlamanız için ayrılan süre, laboratuvarı tamamlamanız için beklenen sürenin 2 katıdır. Zamanlayıcı başladığında, zamanlayıcıyı duraklatamazsınız. Ayrıca, her laboratuvar denemesi önceki denemeyi sıfırlar; yani önceki denemedeki çalışmanız kaydedilmez

**Hedefler**

Bu laboratuvarı tamamladıktan sonra şunları yapabileceksiniz:

• Vitis HLS aracı GUI'sinde yeni bir proje oluşturun

• Kendi kendini kontrol eden bir test tezgahı kullanarak bir C tasarımını simüle edin

• Tasarımı sentezleyin

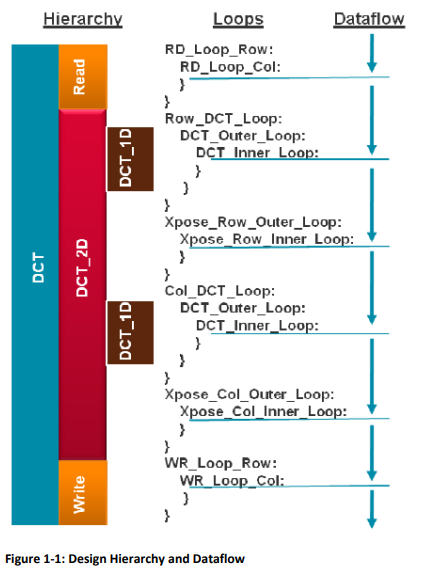
• Analiz Perspektifi görünümünü kullanarak tasarım analizi yapın

• Sağlanan bir C test tezgahı kullanarak oluşturulan bir RTL tasarımı üzerinde ortak simülasyon gerçekleştirin

• Tasarımı uygula

**giriiş**

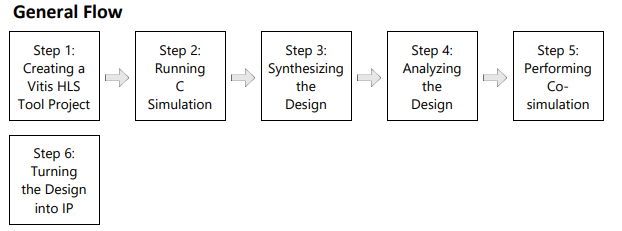
Bu laboratuvar, Vitis High-Level Synthesis (HLS) aracı GUI akışının ana özelliklerini tanıtır. Bir proje oluşturmak için GUI modunda Vitis HLS aracını kullanacaksınız. Ayrıca sağlanan tasarımı simüle edecek, sentezleyecek ve uygulayacaksınız. Bu tasarım ayrık bir kosinüs dönüşümü (DCT) uygular ve C kaynağı olarak sağlanır. İşlev, önce bir 1D DCT aracılığıyla giriş dizisinin her satırını işleyerek, ardından elde edilen dizinin sütunlarını aynı 1D DCT aracılığıyla işleyerek bir 2D DCT algoritmasından yararlanır. read\_data, dct\_2d ve write\_data işlevlerini çağırır. read\_data işlevi 54. satırda tanımlanır ve iki döngüden oluşur: RD\_Loop\_Row ve RD\_Loop\_Col. write\_data işlevi 66. satırda tanımlanır ve sonucun yazılması için iki döngüden oluşur. 23. satırda tanımlanan dct\_2d işlevi, dct\_1d işlevini çağırır ve devrik gerçekleştirir. Son olarak, 4. satırda tanımlanan dct\_1d işlevi, dct\_coeff\_table'ı kullanır ve 1D Type-II DCT algoritmasının temel yinelemeli biçimini uygulayarak gerekli işlevi gerçekleştirir. Aşağıdaki şekil, sol tarafta fonksiyon hiyerarşisini, yürütüldükleri sırayla döngüleri ve sağ tarafta veri akışını göstermektedir.



**Laboratuvar Ortamını Anlamak**

Bu kursta sağlanan laboratuvarlar ve demolar, bir Linux platformunda çalışacak şekilde tasarlanmıştır. Bir ortam değişkeni gereklidir: TRAINING\_PATH, laboratuvar dosyalarının bulunduğu yeri gösterir. Bu değişken, CloudShare/CustEd\_VM ortamlarında yapılandırılmış olarak gelir. Bazı araçlar bu ortam değişkenini doğrudan kullanabilir (yani, $TRAINING\_PATH genişletilecektir) ve bazı araçlar manuel genişletme gerektirir (/home/xilinx/CloudShare/CustEd\_VM ortamları için eğitim). Laboratuvar talimatları, her bir alet için ne yapılması gerektiğini açıklar. Hem Vivado Design Suite hem de Vitis platformu, birçok laboratuvar tarafından kullanılan bir Tcl ortamı sunar. Araç başlatıldığında, araçların önceki başlatılmasından kalan hiçbir işlem veya değişken olmadan temiz bir Tcl ortamıyla başlar.

Bu, bir Tcl komut dosyasını kaynakladıysanız veya herhangi bir Tcl değişkenini manuel olarak ayarladıysanız ve aracı kapattıysanız, aracı yeniden açtığınızda, Tcl komut dosyasını yeniden kaynaklamanız ve laboratuvarın gerektirdiği tüm değişkenleri ayarlamanız gerekeceği anlamına gelir. Bu aynı zamanda terminal pencereleri için de geçerlidir; yeni bir terminal açıldığında tüm değişken ayarlar silinecektir.

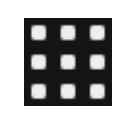


**Bir Vitis HLS Aracı Projesi Oluşturma Adım 1**

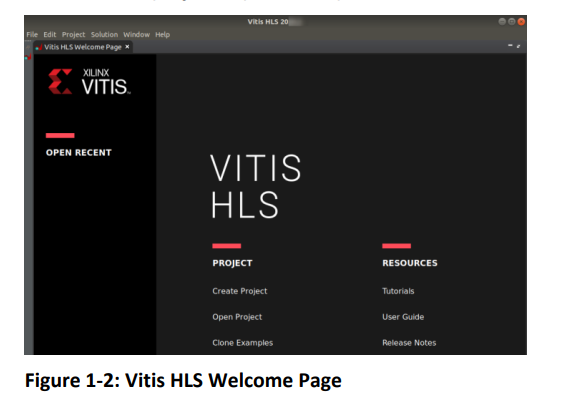
Bu adımda, Vitis HLS aracı GUI'sini başlatacak ve sağlanan C tabanlı ayrık kosinüs dönüşümü (DCT) tasarımı için yeni bir proje oluşturacaksınız. Vitis HLS aracını başlatmanın birkaç yolu vardır. En popüler burada gösterilir

* 1. **Vitis HLS aracını başlatın.**
     1. **Masaüstü veya araç çubuğu simgesini (** **) tıklayın.**

Masaüstü veya araç çubuğu simgesi yoksa, aracı şu şekilde başlatabilirsiniz: [Windows 10 kullanıcıları:] Başlat > Xilinx Tasarım Araçları > Vitis HLS 2022.1'i seçerek.

[VM Linux kullanıcıları:] Araç çubuğundaki () Uygulamaları Göster simgesine tıklayarak, arama penceresine HLS yazarak ve Vitis HLS simgesine tıklayarak.

Vitis HLS aracı, Hoş Geldiniz penceresine açılır. Hoş Geldiniz penceresinden yeni bir proje oluşturabilir, örnekleri açabilir ve belgelere ve örneklere erişebilirsiniz.



Burada sıfırdan yeni bir Vitis HLS aracı projesi oluşturacaksınız.

**1-2**. dct\_prj adlı bir Vitis HLS aracı projesi oluşturun.

**1-2-1.** Karşılama Sayfasından Proje Oluştur'a tıklayın.

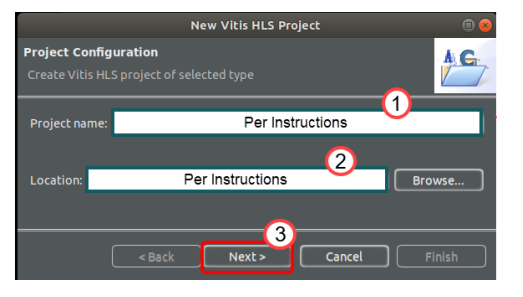
metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Proje Konfigürasyonu iletişim kutusu bir proje adı ve konumu sorar.

**1-2-2.** Proje adı alanına (1) dct\_prj girin.

**1-2-3.** $TRAINING\_PATH/hls\_tool\_flow/lab/[zcu104 | vck190] öğesini Konum alanında (2) seçin.

Laboratuvarı gerçekleştirmek için ZCU104 veya VCK190 hedef kartını seçebilirsiniz.

**1-2-4**. İleri (3)'e tıklayın.

Tasarım Dosyalarını Ekle/Kaldır iletişim kutusu açılır.

1-3. Sağlanan dosyaları projeye ekleyin.

1-3-1. Dosya Ekle'yi tıklayın.

1-3-2. $TRAINING\_PATH/hls\_tool\_flow/support/[zcu104 | vck190]/dct.

1-3-3. Seçmek ve projeye eklemek için dct.c'ye çift tıklayın.

Vitis HLS aracı, çalışma dizinini (proje dizini) ve C dosyalarını içeren herhangi bir dizini arama yoluna otomatik olarak ekler. Bu nedenle, bu dizinlerde bulunan başlık dosyaları otomatik olarak projeye dahil edilir, dolayısıyla bunları açıkça belirtmeye gerek yoktur.

Dosyayı seçip CFLAGS Düzenle'yi tıklatarak (varsa) diğer yollarda bulunan diğer başlık dosyalarının yolunu belirtmelisiniz. Not: Burada her giriş için yeni dosyalar ekleyebilir ve dosyaya özel derleyici yönergelerini listeleyebilirsiniz.

1-4. Araçlara hangi modülün en üst işlev olduğu söylenmelidir. Araç, proje dosyalarını tarayacak ve program hiyerarşisinin en üstünde olacak adayları belirleyecektir. Şimdi bu adaylardan birini seçeceksiniz.

1-4-1. Üst İşlev alanının yanındaki Gözat'a tıklayın. Üst İşlevi Seç iletişim kutusu açılır ve projede mevcut olan aday işlevleri <işlev adı> (<fonksiyonun bulunduğu kaynak dosya>) olarak listeler.

1-4-2. Hangi işlevi "ana" olarak sunmak istediğinizi belirlemek için listeden dct (dct.c) öğesini seçin.

1-4-3. Seçimi kaydetmek için Tamam'a tıklayın.

Not: Üst İşlev alanına üst işlevin adını manuel olarak da girebilirsiniz.

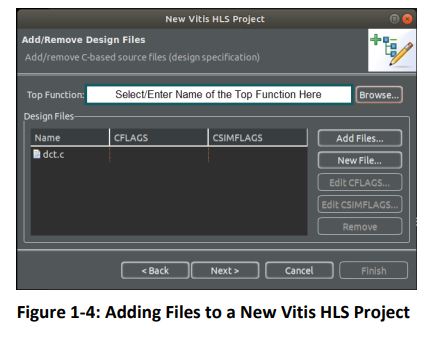
C programları genellikle ana () işlevini üst düzey işlev olarak kullanır. Vitis HLS aracı tasarım akışı, sentez için en üst düzey işlev olarak main() altındaki herhangi bir alt işlevin kullanılmasını destekler. Üst düzey main() işlevini sentezleyemezsiniz.

Aşağıdakiler ek kurallardır:

o Sentez için en üst düzey işlev olarak yalnızca bir işleve izin verilir.

o Üst düzey fonksiyon ve altındaki tüm fonksiyonlar bir birim olarak sentezlenir.

o Üst düzey işlevin altındaki hiyerarşide olmayan ek işlevler, sentez için üst düzey işlevle birleştirilmelidir.



1-4-4. Bu dosyaları projeye eklemek ve test tezgahı dosyaları sayfasına ilerlemek için İleri'ye tıklayın.

1-5. Test tezgahı dosyalarını ekleyin. Test tezgahı dosyaları burada belirtilir.

1-5-1. Dosya Ekle'yi tıklayın.

1-5-2. $TRAINING\_PATH/hls\_tool\_flow/support/[zcu104 | vck190]/dct.

1-5-3. dct\_test.c, in.dat, out.golden.dat öğesini seçin.

1-5-4. Bu dosyaları eklemek için Aç'ı tıklayın.

1-5-5. Çözüm Yapılandırma sayfasına ilerlemek için İleri'ye tıklayın.

1-6. Tasarımın bazı fiziksel parametrelerini belirtin.

Fiziksel parametreler saat hızı, parça seçimi ve diğer seçenekleri içerir.

Varsayılan olarak, Çözüm Adı alanında çözüm1 doldurulur ve istenirse geçersiz kılınabilir.

1-6-1. Saat periyodunu 10'a ayarlayın.

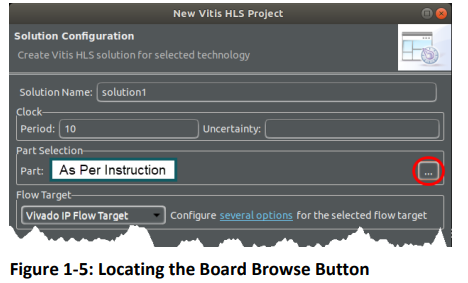
Not: Saat periyodu için varsayılan birim nanosaniyedir.

Belirsizlik alanı, uygulanabilir bir sonuç elde etmek için araçların daha fazla çalışmasına neden olacak bir katılık (kısıtlama) belirlemenizi sağlar.

Hızlı bir hatırlatma olarak, saat belirsizliği, yerel bir saat ağındaki zaman farkı, yükselen kenarın bir register'a en erken varış zamanı ile yükselen kenarın herhangi bir register'a en geç varış zamanı arasındaki zaman farkıdır. Bu sayı bazen çarpık olarak adlandırılır.

1-6-2. Belirsizlik alanını boş bırakın.

1-6-3. Bir parça veya pano seçmek için Parça Seçimi seçeneğinden Gözat (. . .) simgesine tıklayın.



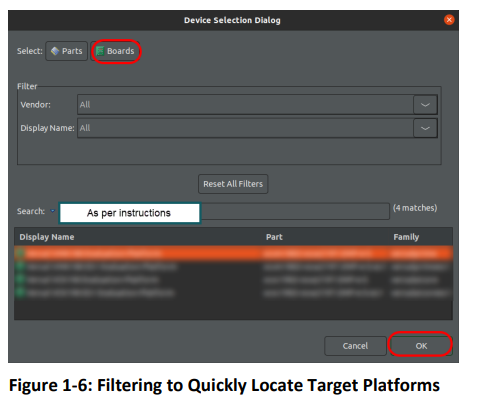
Cihaz Seçimi İletişim kutusu açılır

1-6-4. Seçiminizi bir parça ataması belirtmek yerine pano adına göre yapmak için Panolar'a tıklayın.

1-6-5. Arama alanına ZYNQ UltraScale+ veya Versal yazın.

Alternatif olarak, çok daha hızlı olan ZCU104 veya VCK190 gibi adın bir bölümünü yazabilirsiniz.

Not: 2022.1 için sonuçların görünmeyebileceği bilinen bir sorun vardır. Bunun hızlı çözümü Parçalar düğmesine ve ardından Panolar düğmesine tıklamaktır.



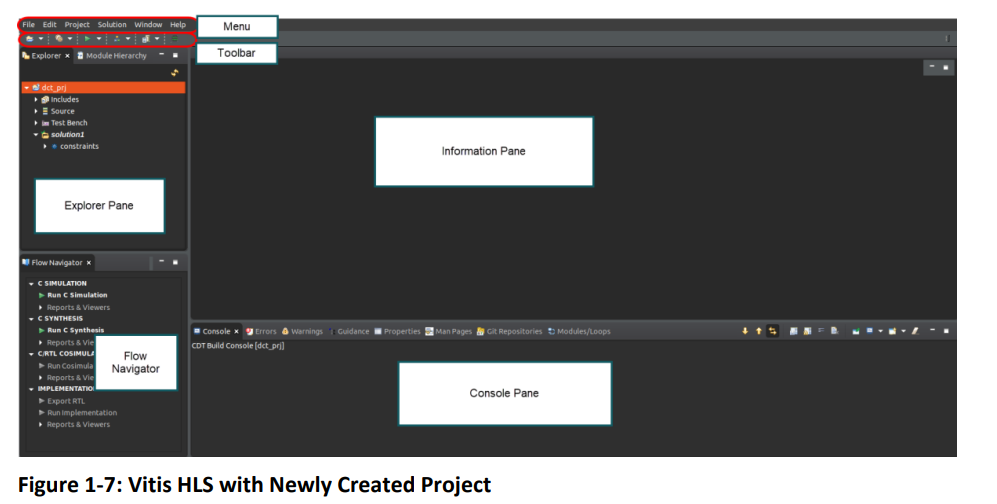
1-6-6. Arama listesinden ZYNQ UltraScale+ ZCU104 Değerlendirme Kurulu veya Versal VCK190 Değerlendirme Platformu'nu seçin.

1-6-7. Panoyu seçmek için Tamam'a tıklayın.

1-6-8. Henüz ayarlanmadıysa, Akış Hedefi alanı altındaki açılır listeden Vivado IP Akış Hedefi'ni seçin.

1-6-9. Bitir'i tıklayın.

Oluşturulan projeyi Explorer sekmesinde göreceksiniz.



Vitis HLS aracı GUI, geliştirme çalışmasına devam etmek için çeşitli bölmelerden oluşur:

▪ Gezgin görünümü, proje hiyerarşisinde gezinmenizi sağlar. Diskteki proje dizininde de benzer bir hiyerarşi vardır. Genişletilebilir alt klasörler, kaynak dosyalar ve test tezgahları gibi çeşitli proje bileşenlerini düzenler.

▪ Bilgi alanı, rapor özetlerini görüntüler ve açık dosyaların içeriğini gösterir. Dosyalar, Explorer görünümünde çift tıklanarak açılabilir.

▪ Konsol görünümü, Vitis HLS aracı sentez veya simülasyon çalıştırırken çıktıyı görüntüler.

▪ Akış Gezgini görünümü, simülasyon, sentez ve dışa aktarılan çıktı yoluyla kaynak kodunuzu almak için komutlara ve işlemlere erişim sağlar.

▪ Bilgi alanında kaynak kodun ne zaman açıldığı ve Anahat ve Yönerge görünümlerinin sağ tarafta görüntülendiği ve kodun hiyerarşisi ile ilgili bilgileri gösterdiği gibi, gerçekleştirilen işlemlere bağlı olarak diğer görünümler dinamik olarak açılır.

**C Simülasyonu Çalıştırma Adım 2**

Oluşturulan proje ile artık hem sözdizimi hem de davranış için C kodunu doğrulayacaksınız. Davranışı doğrulamak için sağlanan kendi kendini kontrol eden C test tezgahı kodunu kullanacaksınız.

Kısa bir gözden geçirme olarak: Ayrık kosinüs dönüşüm algoritması, farklı frekans kosinüs fonksiyonlarının toplamı cinsinden sonlu bir veri noktaları dizisini ifade eder.

DCT, bir görüntüyü spektral alt bantlara ayırmak için yaygın olarak kullanılır. FFT veya DFT gibi, algoritmada gerçekleştirilen birçok yineleme vardır.

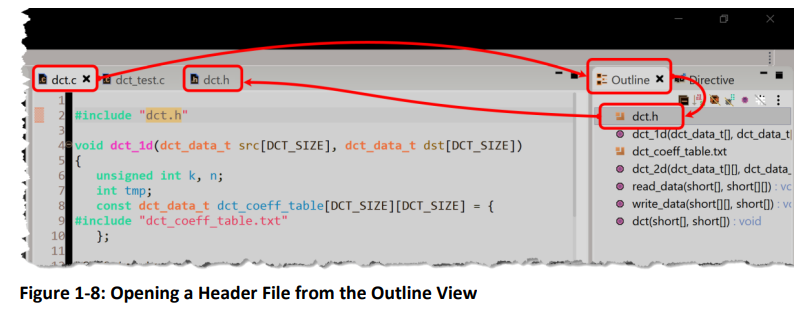
2-1. Sağlanan kaynak ve test karşılaştırma dosyalarını inceleyin.

2-1-1. Gezgin bölmesinde Kaynak klasörünü genişletin.

2-1-2. Dosyayı açmak için dct.c'ye çift tıklayın. Bu, bilgi bölmesinde kaynak dosyayı açacaktır.

2-1-3. Kod yapılarını gözden geçirin.

2-1-4. Bu başlık dosyasını yeni bir düzenleyici penceresinde açmak için Anahat görünümünden dct.h öğesine çift tıklayın.



2-1-5. Başlık dosyasının içeriğini gözden geçirin.

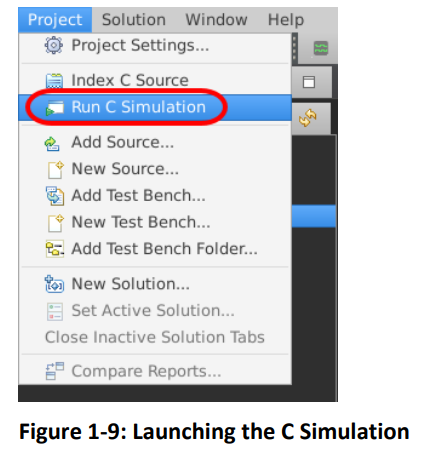
2-1-6. Gezgin bölmesinde Test Bench klasörünü genişletin.

2-1-7. Bilgi bölmesinde açmak için dct\_test.c'ye çift tıklayın.

Bu test tezgahı, kendi kendini kontrol eden bir test tezgahıdır; yani, hesaplanan çıktı bir referans altın çıktısıyla karşılaştırılır ve başarılı veya başarısız olarak döner.

**2-2.** Vitis HLS araç tasarımını simüle edin.

2-2-1. Proje > C Simulation'ı Çalıştır'ı seçin.

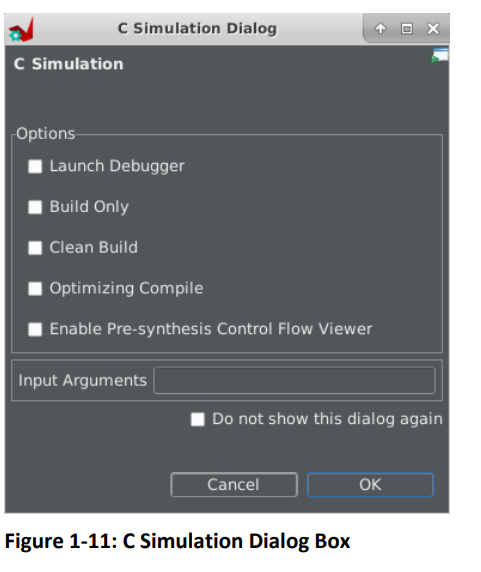


C Simülasyonu altında C Simülasyonunu Çalıştır'ı tıklatarak Flow Navigator'dan C simülasyonunu da çalıştırabilirsiniz.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

C Simulation'ı Çalıştır iletişim kutusu açılır.



Seçeneklerin her biri simülasyonun nasıl çalıştırılacağını kontrol eder:

o Hata Ayıklayıcıyı Başlat: Derlemeden sonra, kodda adım adım ilerlemeniz için hata ayıklama perspektifi otomatik olarak açılır.

o Yalnızca Oluştur: Kaynak kodunu ve test tezgahını derler, ancak simülasyon çalışmaz. Bu seçenek, simülasyonu çalıştırmadan önce derleme sürecini test etmek ve derlemeyle ilgili sorunları çözmek için kullanılabilir.

o Temiz Yapı: Kodu derlemeden önce mevcut tüm yürütülebilir dosyaları ve nesne dosyalarını kaldırır.

o Derlemeyi Optimize Etme: Varsayılan olarak, tasarım hata ayıklama bilgileri etkinleştirilmiş olarak derlenir ve derlemenin analiz edilmesine ve hatalarının ayıklanmasına izin verir. Derlemeyi İyileştirme seçeneği, tasarımı derlerken daha yüksek düzeyde bir optimizasyon çabası kullanır ancak hata ayıklayıcının gerektirdiği bilgileri eklemez. Bu, derleme süresini artırır ancak simülasyon çalışma süresini azaltmalıdır.

o Sentez Öncesi Kontrol Akış Görüntüleyicisini Etkinleştir: Sentez Öncesi Kontrol Akışı raporunu oluşturur.

o Giriş Bağımsız Değişkenleri: Test tezgahı main() işlevinizin gerektirdiği tüm girişleri belirtin.

2-2-2. Varsayılan Seçenekler'i seçin (yani, bir C simülasyonunu çalıştırmak için ek seçim gerekmez).

2-2-3. Tamam'ı tıklayın.

Simülasyon günlüğü düzenleyici bölmesinde görüntülenecektir.

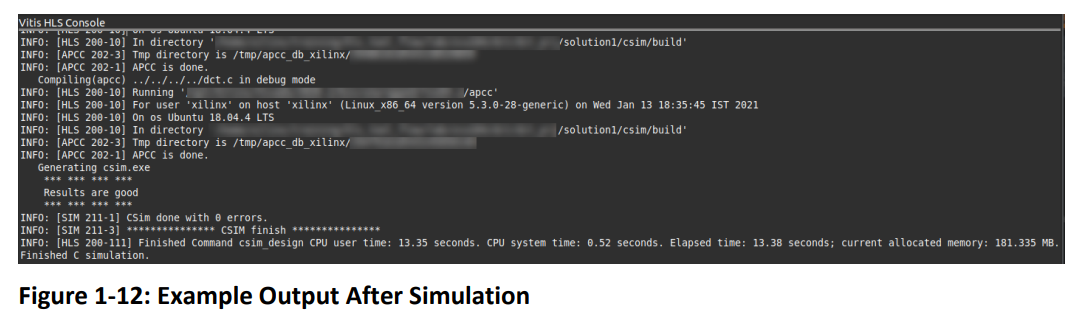
2-3. Simülasyon raporunu görüntüleyin.

Vitis HLS aracı tarafından oluşturulan bilgiler, her ikisi de burada açıklanan iki yerde bulunabilir.

Birincisi, yalnızca kod tarafından üretilen çıktıyı değil, aynı zamanda simülasyon motoru mesajlarının tamamını da bildiren Konsol penceresidir. Simülasyon günlüğü, yalnızca temel simülasyon motoru mesajlarını ve simüle edilmiş kod çıktısını sağlar.

2-3-1. Aracın GUI'sinin alt kısmındaki Konsol sekmesini seçin.

Simülasyon tarafından üretilen tüm çıktıları görüntülemek için kaydırmanız gerekebilir.



Aşağıda açıklanan diğer konum, yalnızca birkaç simülasyon motoru mesajı ve simüle edilmiş kod çıktısı sağlar. Tipik olarak bu, simülasyon tamamlandıktan sonra açılır; ancak, günlük bölmesini kapattıktan sonra buna erişmeniz gerekirse, simülasyon raporuna nasıl erişeceğiniz aşağıda açıklanmıştır.

2-3-2. Gezgin bölmesinde dct\_prj > solution1 > csim > rapor öğesini genişletin.

2-3-3. Düzenleyici bölmesinde açmak için günlük dosyası adına çift tıklayın.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Simülasyon günlük dosyasında ve konsol alanında "Sonuçlar iyi" mesajını görmelisiniz.

**Tasarım Aşamasının Sentezlenmesi adım 3**

Ardından, varsayılan ayarları kullanarak tasarımı sentezleyeceksiniz. Ardından, C işlevini uygulamak için gereken kaynakları bulmak için sonucu analiz edeceksiniz.

3-1. Tasarımı sentezleyin.

3-1-1. C sentez aracını başlatmak için aşağıdaki yöntemlerden birini kullanın:

o Araç çubuğunda (1) Akışı Çalıştır simgesine ( ) tıklayın.

o Araç çubuğundaki Run Flow simgesinin yanındaki açılır menüye tıklayın ve C Synthesis(2)'yi seçin.

o Explorer sekmesinden sentezlemek istediğiniz çözümü seçin ve sağ tıklayın ve C Sentez > Aktif Çözüm (3) öğesini seçin.

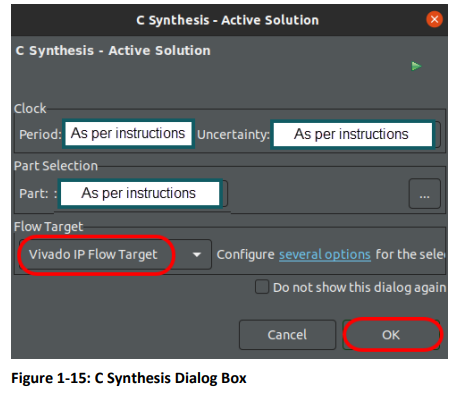
Not: Daha karmaşık projeler, bu menüdeki diğer seçeneklerin desteklediği birden fazla çözümün gruplanmasına izin verir.

o Flow Navigator'dan, C Synthesis > Run C Synthesis'i (4) seçin.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

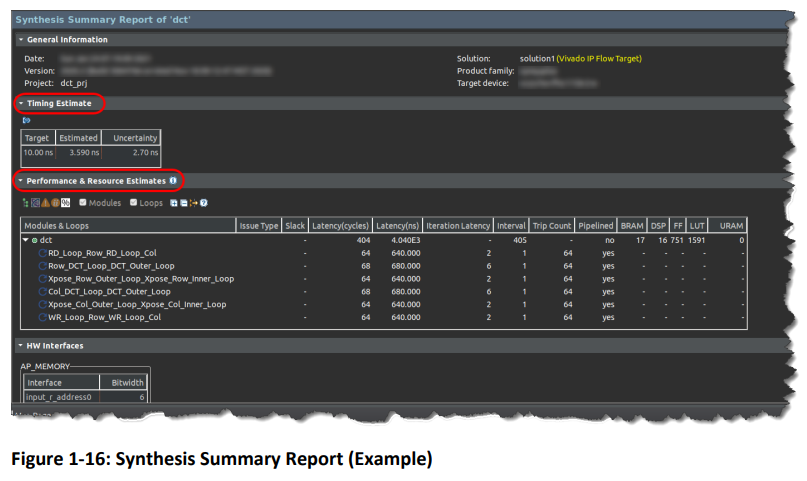
C Sentezi - Aktif Çözüm iletişim kutusu açılır.

Araç, projenin saat periyodu, parça seçimi ve akış türü gibi zaten bildiği tüm bilgileri otomatik olarak doldurur. Gerekirse bunları burada değiştirebilirsiniz.

3-1-2. Sentez sürecini başlatmak için Tamam'a tıklayın.

Not: Akış hedefinin Vivado IP Akış Hedefi olarak seçilmesi, araçların Vivado Design Suite'te kullanılmak üzere RTL IP dosyaları oluşturmasına neden olur. Daha basit bir ifadeyle, burada oluşturacağınız ve Vivado Design Suite'e aktaracağınız daha büyük bir tasarımda kullanılmak üzere bir IP parçası oluşturuyorsunuz. Bu aracı kullanmanın başka bir yolu da akış hedefi olarak Vitis Kernel Flow'u seçmektir. Vitis uygulama hızlandırma akışı için derlenmiş bir çekirdek nesnesi (.xo). Bu daha dar bir çözümdür ve diğer laboratuvarlarda tartışılmaktadır.

Sentez tamamlandığında, Bilgi bölmesinde Sentez Özet Raporu görünür. Bu çözümün çalıştırılabileceği tahmini saat periyodunu gösteren Zamanlama Tahmini bölümüne dikkat edin. Belirsizlik faktörünü içeren tahmini süre, hedef değerden küçük olduğu sürece, tasarım zamanlamayı karşılayacaktır. Performans ve Kaynak Tahminleri bölümü, bu tasarımı uygulamak için gereken toplam kaynak sayısını gösterir. Üst düzey bu kaynakları gösterir ve koddaki bileşen öğeleri aşağıda listelenmiştir. Toplam ve bileşen gecikmeleri, sistem performansının belirlenmesine yardımcı olmak için sağlanır.



3-1-3. Gezgin bölmesinde dct\_prj > solution1 > syn > rapor klasörünü genişletin.

3-1-4. Sentez raporunu görüntülemek için dct\_csynth.rpt dosyasına çift tıklayın.

3-1-5. Aşağıdaki soruyu yanıtlamak için Sentez raporunda Performans Tahminleri ve Kullanım Tahminleri'ne gidin.

**Soru 1**

**Sentez raporundan aşağıdaki ayrıntıları yazın:**

**• Tahmini saat frekansı: 3.590ns**

**• En kötü durum gecikmesi: 415**

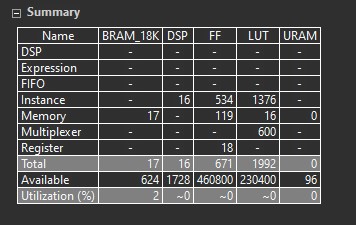
**• BRAM\_18K sayısı: 17**

**• Kullanılan DSP48E sayısı: 16**

**• Kullanılan FF sayısı: 671**

**• Kullanılan LUT sayısı: 1992**

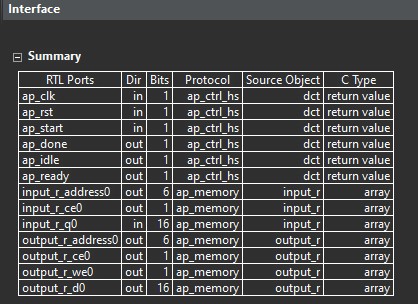
**URAM : 0**

**metin içeren bir resim

Açıklama otomatik olarak oluşturuldu**

3-1-6. Sentez raporunda Arayüz > Özet'i seçin.

Bu rapor, araçlar tarafından oluşturulan üst düzey arayüz sinyallerini gösterir.



ap\_clk ve ap\_rst sinyalleri otomatik olarak eklenir.

ap\_start, ap\_done ve ap\_idle, tasarımın bir sonraki hesaplama komutunu (ap\_idle), bir sonraki hesaplamanın ne zaman başlatıldığını (ap\_start) ve hesaplamanın ne zaman tamamlandığını (ap\_done) belirtmek için anlaşma sinyalleri olarak kullanılan üst düzey sinyallerdir. Diğer sinyaller tasarımın kendisine dayalı olarak üretilir.

3-1-7. Konsol sekmesini seçin.

Sentez günlüğü, Vitis HLS Konsolunda mevcuttur.

Solution1 > syn klasörü Explorer görünümünde genişletildiğinde, rapor dosyalarının ve oluşturulan kaynak dosyalarının (VHDL, Verilog, başlık ve cpp) altında bulunduğu rapor, verilog ve vhdl alt klasörlerini göstereceğini unutmayın. Bu girişlerden herhangi birine çift tıklamak, Bilgi bölmesinde ilgili dosyayı açacaktır. Ortaya çıkan tasarım, işlevler kullanılarak hiyerarşik olarak yapılandırılır ve alt düzey işlevlere karşılık gelen raporlar oluşturulur. Varsayılan olarak, üst düzey işlev raporu, sentez tamamlandıktan sonra Bilgi bölmesinde görüntülenir.

**Oluşturulan Raporları Kullanarak Tasarımın Analiz Edilmesi Adım 4**

Artık sentez aşaması tamamlandığında, uygulamanın performansını anlamanıza ve analiz etmenize yardımcı olmak için sentez raporlarını daha ayrıntılı olarak analiz edeceksiniz. Bu raporlar Sentez Özeti raporunu, Zamanlama Görüntüleyicisini, İşlev Çağrısı Grafiği'ni ve Veri Akışı Görüntüleyicisini içerir. Bu raporlara Akış Gezgini'nden erişilir:

o Program Görüntüleyici: Fonksiyonun her bir işlem ve kontrol adımını ve içinde gerçekleştirdiği saat döngüsünü gösterir.

o Veri Akışı Görüntüleyici: Aracın çıkardığı veri akışı yapısını göstererek kanalları (FIFO/PIPO) incelemenize ve kanal derinliğinin performans üzerindeki etkisini incelemenize olanak tanır.

o İşlev Çağrısı Grafiği Görüntüleyici: Gecikme ve başlatma aralığı (II) açısından tasarımın verimini göstermek için C sentezi veya C/RTL ortak simülasyonundan sonra tam tasarımınızı görüntüler.

Varsayılan olarak, Zamanlama Görüntüleyici görüntülenir. Tasarım performansı ve kaynaklarının hem tablo hem de grafik görünümünü sağlar.

4-1. Analiz perspektifine geçin ve davranışı Çizelge Görüntüleyici aracılığıyla görüntüleyin.

4-1-1. Zamanlama Görüntüleyici'yi açmak için Çözüm > Zamanlama Görüntüleyiciyi Aç'ı seçin. Alternatif olarak, Akış Gezgini'nden C Sentez > Raporlar ve Görüntüleyiciler'i genişletin ve Zamanlama Görüntüleyici'ye tıklayın. Zamanlama Görüntüleyici açıldığında, aşağıdaki pencereler görünür:

o Zamanlama Görüntüleyici: Verilerin tasarımda nasıl hareket ettiğini grafik olarak gösterir.

o Modül Hiyerarşisi: Mevcut hiyerarşinin fonksiyon hiyerarşisini ve performans özelliklerini gösterir.

o Modüller/Döngüler: Raporun altındaki Modüller/Döngüler tablosunda her bir işlev/döngü için ek performans ve kaynak ölçümlerini gösterir.

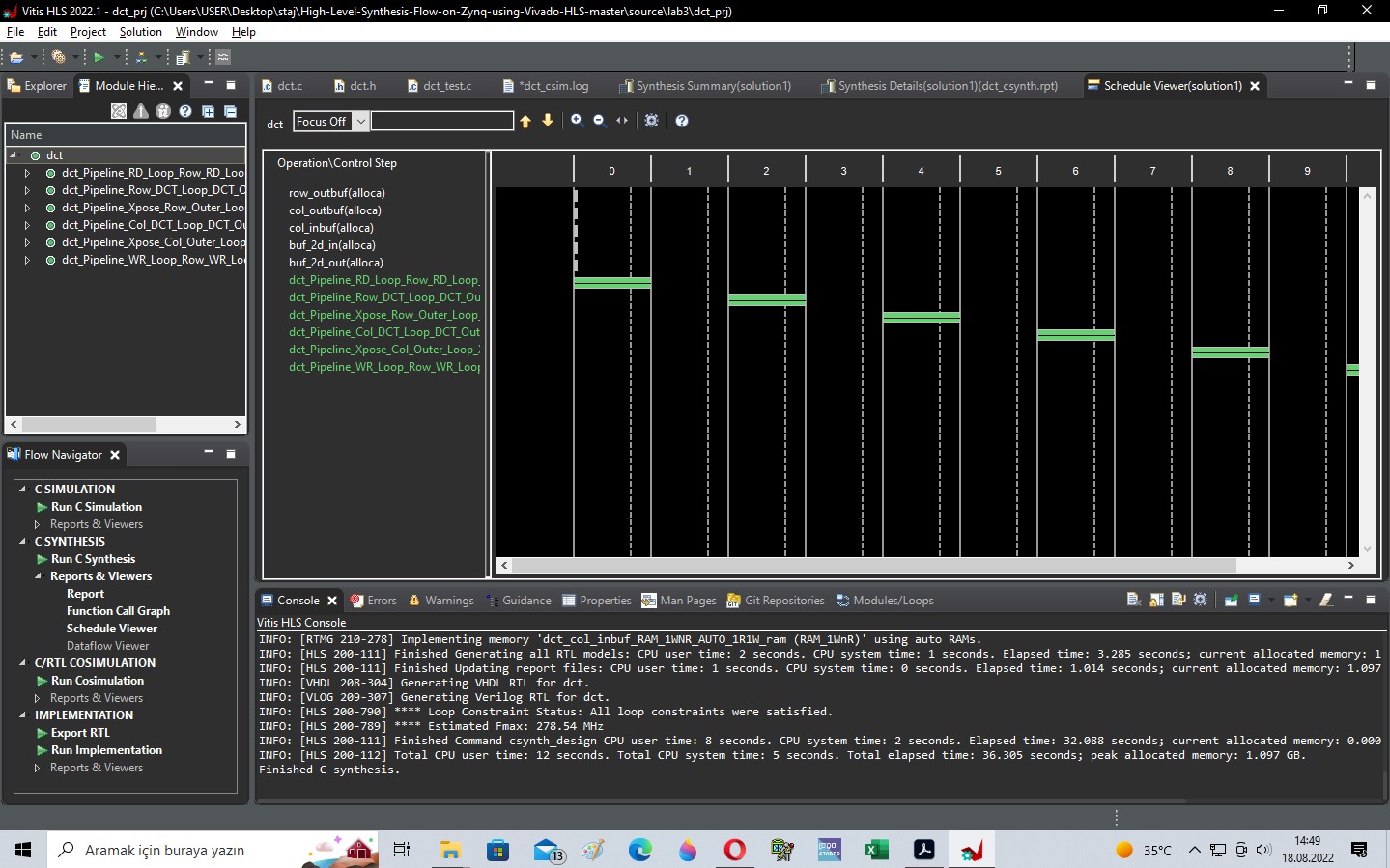
o Performans Profili: Herhangi bir performans bilgisi olmadan üst düzey fonksiyondan döngüleri gösterir.

o Kaynak Profili: Sentezlenen işlevin farklı öğelerinin kaynak kullanımını gösterir.

o Özellikler görünümü: Şu anda seçili olan kontrol adımının veya işlemin özelliklerini Zamanlama Görüntüleyici'de gösterir.

Modül Hiyerarşisi bölmesi, tüm RTL tasarımına genel bir bakış sağlar. RTL hiyerarşisindeki her blok için kaynakları ve gecikme katkısını gösterir. Bu görünüm, herhangi bir II veya zamanlama ihlallerini doğrudan gösterir. Zamanlama ihlalleri durumunda, hiyerarşi penceresi ayrıca belirli bir modülde gözlemlenen toplam negatif bolluğu da gösterecektir.

Modül hiyerarşilerinin varsayılan olarak genişletilmemiş olarak görüntülendiğini unutmayın.

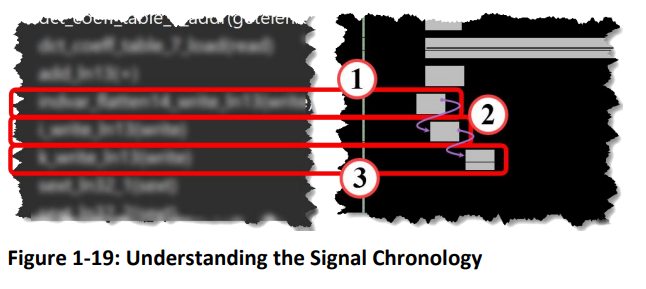
Zamanlama Görüntüleyici, işlev veya döngünün her bir işlemi ve kontrol adımını ve yürüttüğü saat 

Bu da schedule viewer ekranı

döngüsünü gösteren veri akışını gösterir. Zamanlama Görüntüleyici, paralelliği önleyen ve zamanlama ihlallerine ve veri bağımlılıklarını tespit etmeye neden olan döngü bağımlılıklarının belirlenmesine yardımcı olur.

Sol dikey eksen, RTL hiyerarşisindeki operasyonların ve döngülerin adlarını topolojik sırayla gösterir ve bu satırdaki bir işlemin yalnızca yürütülebileceğini gösterir. önceki hattan işlemlerle ve sonraki (sonraki) hatları çalıştırabilir.

Örnek olarak, sinyal 1, sinyal 2'den önce gerçekleşir ve bu nedenle daha önce, yani sinyal listesinde daha yüksek görünür. Sinyal 3, sinyal 2'yi takip eder ve sinyal 2'den sonra veya sinyalin altında görünür.



Üst yatay eksen, saat döngülerini ardışık sırada gösterir.

Her saat döngüsündeki dikey kesikli çizgi, saat belirsizliğinden dolayı saat periyodunun ayrılmış kısmını gösterir. Bu süre, araç tarafından yer ve rota işlemleri ve diğer arka uç görevleri için bırakılır.

Her işlem tabloda bir kutu olarak gösterilmiştir. Kutu, toplam saat döngüsünün yüzdesi olarak işlemin gecikmesine göre yatay olarak boyutlandırılmıştır. İşlev çağrıları durumunda, sağlanan döngü bilgisi, işlem gecikmesine eşdeğerdir. Çok döngülü işlemler, kutunun ortasından geçen yatay bir çizgi ile kutular olarak gösterilir. Genel operatör veri bağımlılıkları da düz mavi çizgiler olarak görüntülenir. Yeşil noktalı çizgi, yinelemeler arası veri bağımlılığını gösterir. Bellek bağımlılıkları altın çizgiler kullanılarak görüntülenir. Kaynak kodu, Zamanlama Görüntüleyici raporundaki her işlemle ilişkilendirilir. İşlemle ilişkili kaynak kodunu ayrı bir sekmede açan Kaynağa Git komutuna erişmek için işlemi sağ tıklayın.

4-2. Zamanlama Görüntüleyici özelliğiyle dct modülünün performansını analiz edin. dct modülünün üç ana kaynağı vardır:

• Hesaplamada kullanılan verileri input.txt dosyasından okuyan RD\_Loop\_Row adlı bir döngü.

• dct\_2d adlı bir alt blok.

• Sonuçları output.txt dosyasına yazan WR\_Loop\_Row adlı bir döngü.

Tasarımdaki tüm döngülerin varsayılan olarak ardışık düzende olduğuna dikkat edin.

4-2-1. Bağlam menüsünü açmak için Modül Hiyerarşisinde dct\_Pipeline\_RD\_Loop\_Row\_RD\_Loop\_Col öğesine sağ tıklayın.

4-2-2. Bu modüle odaklanan Zamanlama Görüntüleyici'yi açmak için Zamanlama Görüntüleyiciyi Aç'ı seçin.

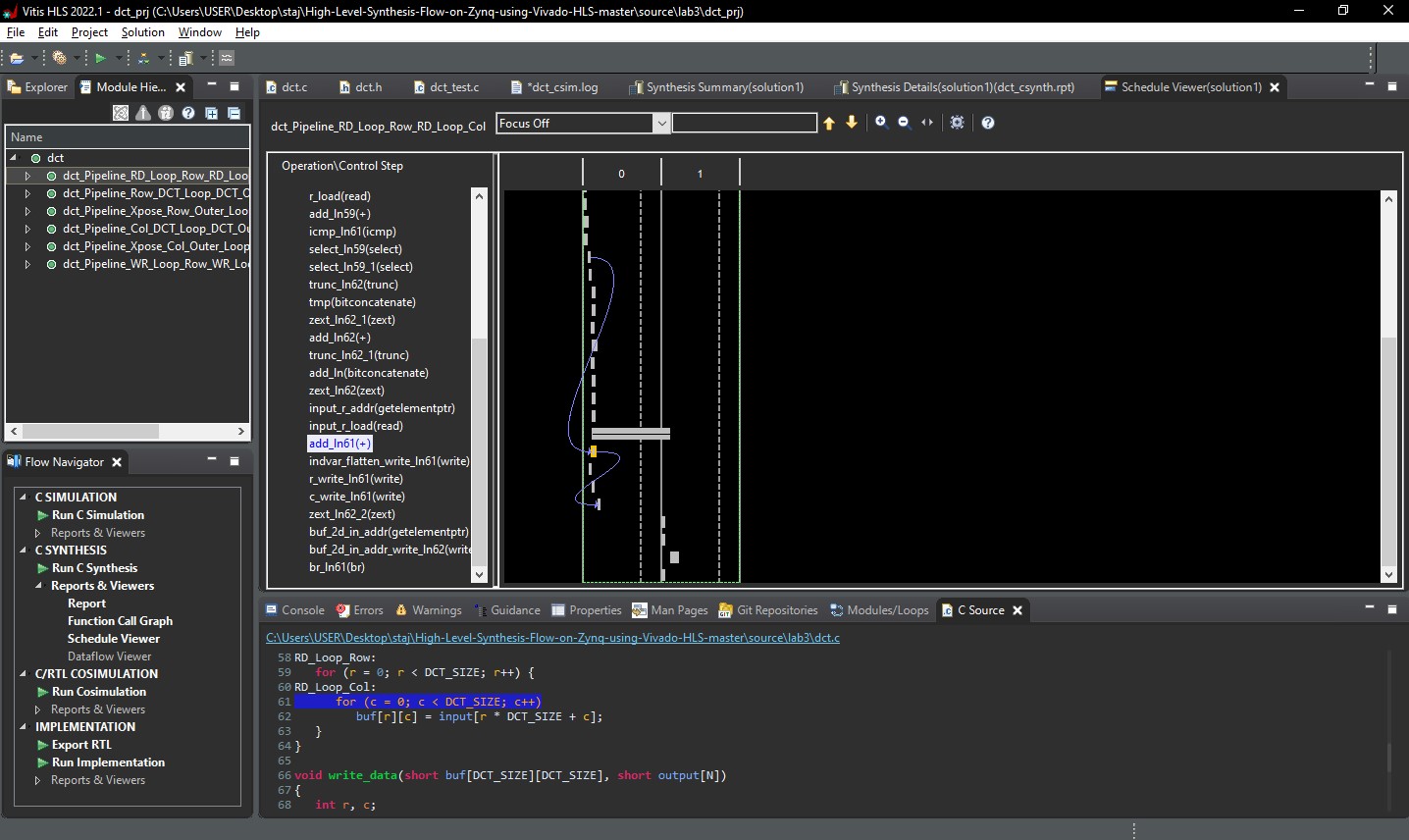
4-2-3. görmek için Zamanlama Görüntüleyicide RD\_Loop\_Row\_RD\_Loop\_Col öğesini genişletin.

RD\_Loop\_Row\_RD\_loop\_Col girişi.Bu, Schedule Viewer'daki görünümün değişmesine ve bu hiyerarşi düzeyindeki sinyalleri göstermesine neden olur.

4-2-4. Hem bu girişin grafiğini vurgulamak hem de bağlam menüsünü açmak için add\_In61(+) öğesine sağ tıklayın.

Not: Bu sinyal, tanık olmak üzere olduğunuz bazı işlevleri göstermek için rastgele seçilmiştir.

4-2-5. İlgili C kaynak kodunu görmek için Kaynağa Git'i seçin.



Zamanlama Görüntüleyici'de sunulan bilgiler, yürütülecek ilk kaynak kümesi gözden geçirilerek aşağıda açıklanmıştır:

o Tasarım 0 durumunda başlar.

o Daha sonra RD\_Loop\_Row döngüsündeki mantığı yürütmeye başlar.

▪ Not: Döngünün ilk durumunda çıkış koşulu kontrol edilir ve bir ekleme işlemi vardır.

o Döngü iki durum üzerinde yürütülür: 0, 1

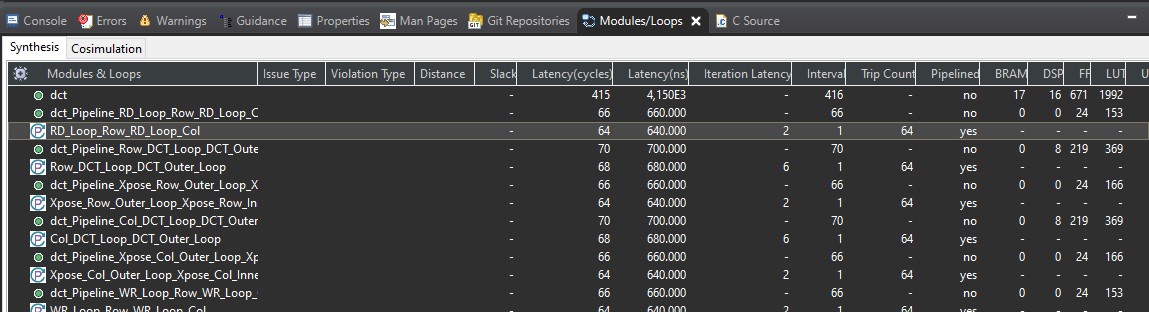
**soru 2**

**Mor dalgalı çizgi neyi temsil ediyor?**

Düz çizgiler arasında olan işlemler bir cycle boyunca yapılan işlemleri belirtir noktalılar ise belirsizdir

Bloğun genişliği, gecikmeyi gösterir ve bloklar arasındaki geçişe kadar olan eğrilik, yönlendirmeden kaynaklanan gecikmeyi gösterir. Source kod ile dalgalı çizginin rengi aynı ise for döngüsününde sütün artısı c++ olduğunu belirtir

4-2-6. Bu bölme bilgilerinde kullanılan performansı ve kaynakları gözden geçirin.



Bu bölme, bu döngünün 64'lük bir açma sayısına sahip olduğunu gösterir. Bu nedenle, bu üç durum boyunca iki kez yinelenir. RD\_Loop\_Row döngüsünün yürütülmesi için 64 saat döngüsü sürdüğünü gösterir. RD\_Loop\_Col döngüsü içinde, bazı toplayıcılar, iki döngülü bir okuma işlemi ve bir yazma işlemi olduğunu görebilirsiniz. Bu, hem döngü için gerekli bir toplayıcı hem de bir döngü kontrol mekanizması ve biraz matematik olduğu için mantıklı olmalıdır. hem buf hem de girdi dizilerinin dizinlerini yönetmek için gerçekleştirilmelidir. Bu bölme, bu hiyerarşi düzeyinde kullanılan kaynakları da gösterir. Bu tasarımda şunları yapabilirsiniz:

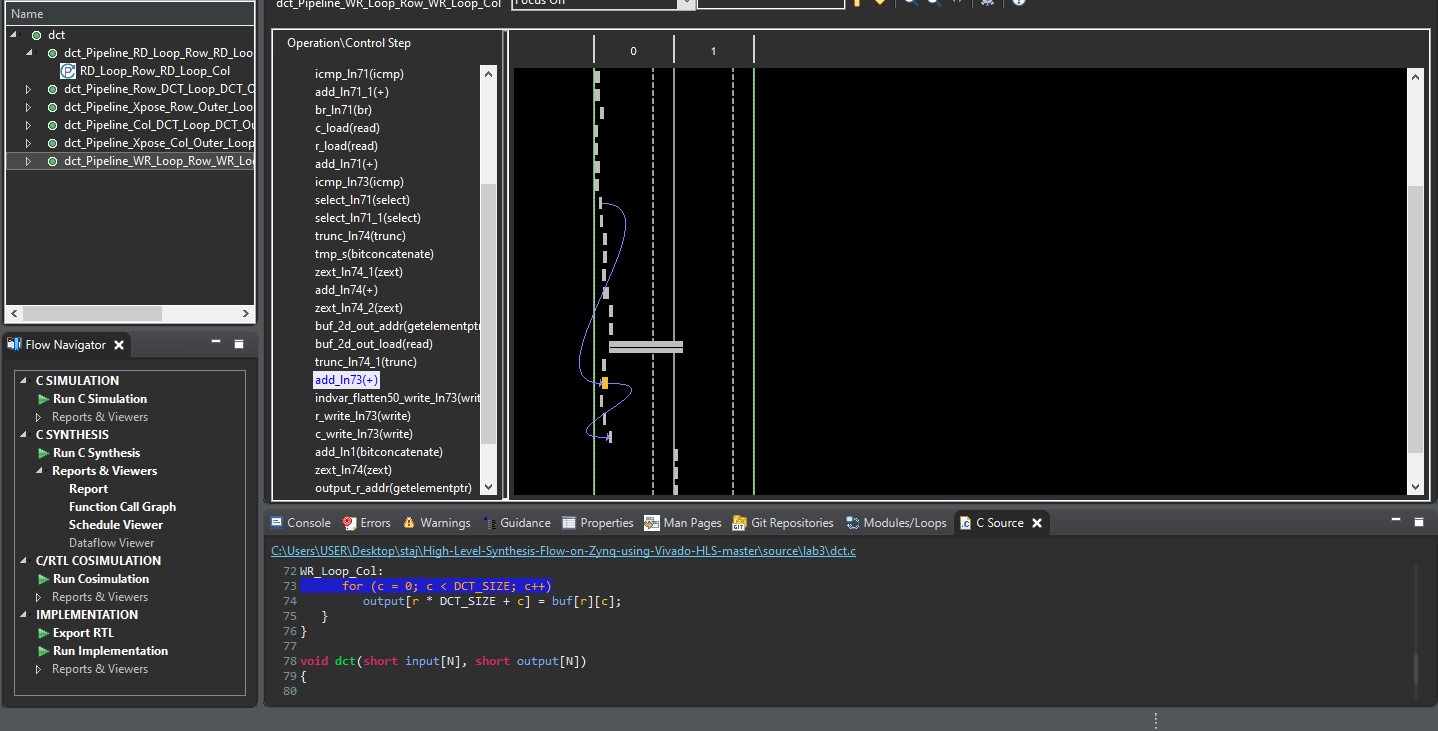
kaynakların çoğunun, bu blok içinde somutlaştırılan bloklar olan örneklere bağlı olduğunu görün. Modül Hiyerarşisi sekmesinde seçilen fonksiyona göre, o fonksiyon tarafından kullanılan kaynakları görüntüleyebilirsiniz.

4-2-7. Modül Hiyerarşisinden dct\_Pipeline\_WR\_Loop\_Row\_WR\_Loop\_Col öğesini seçin.

4-2-8. Zamanlama Görüntüleyici'de (çözüm1) WR\_Loop\_Row\_WR\_Loop\_Col döngüsünü genişletin.

4-2-9. Hem sinyali seçmek hem de bağlam menüsünü açmak için add\_In73(+) öğesine sağ tıklayın.

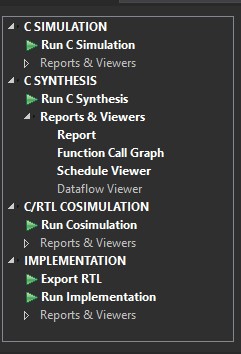
4-2-10. İlgili C kaynak kodunu görmek için Kaynağa Git'i seçin.



Bu yazma işlemi, verileri buf dizisinden çıktı dizisi değişkenine kopyalamaktır.

4-3. İşlev Çağrısı Grafiği özelliğini kullanarak tasarım analizine devam edin.

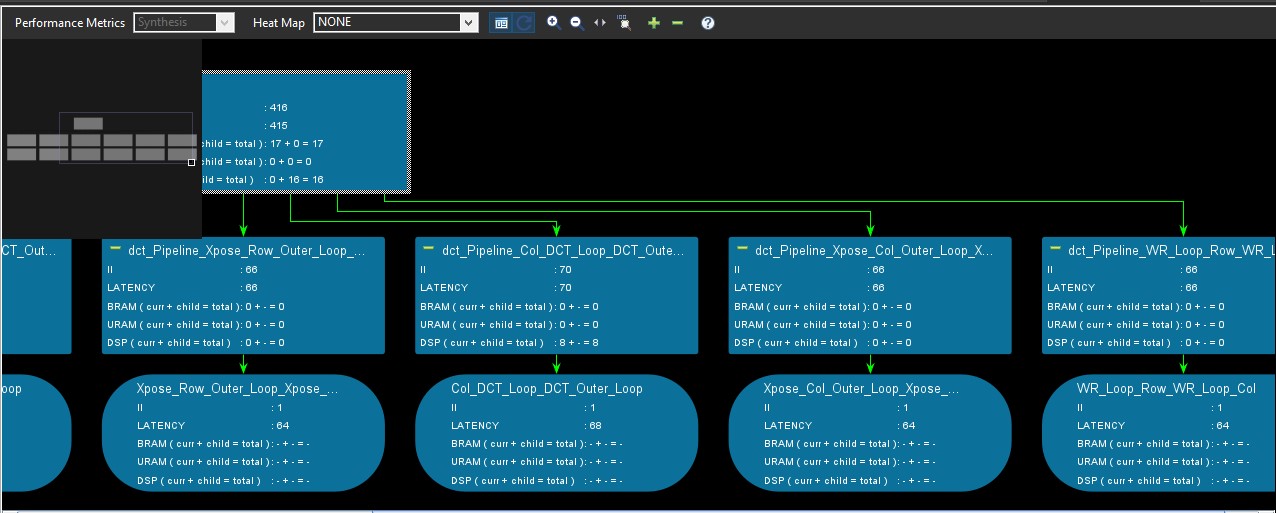
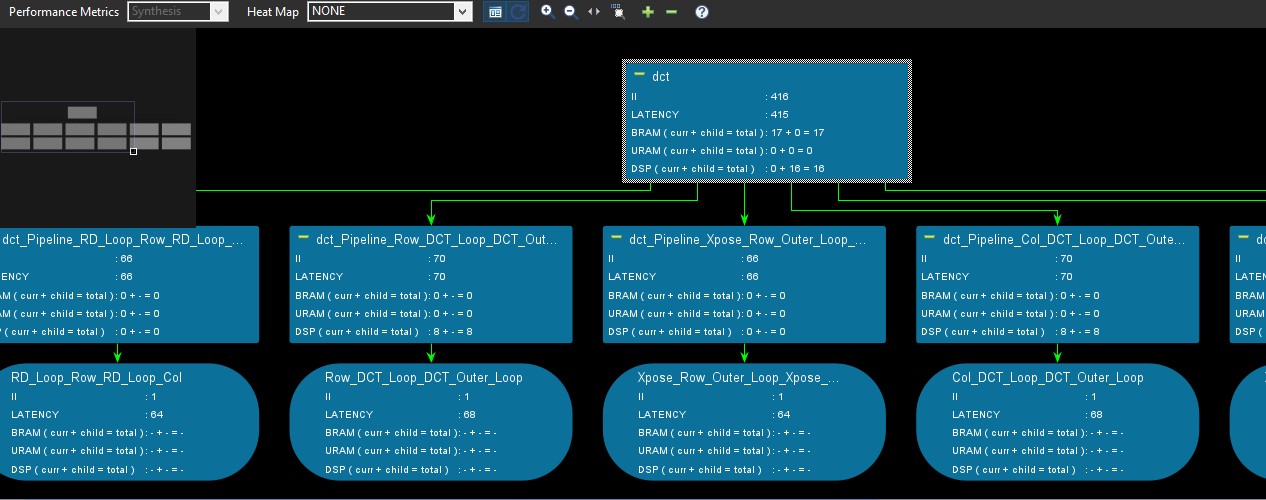
4-3-1. Akış Gezgini'nden İşlev Çağrısı Grafiği'ni seçin.



Bu, İşlev Çağrısı Grafiği Görüntüleyicisini açar ve C sentezinden sonra gecikme ve başlatma aralığı (II) açısından tasarımın verimini gösterir.

İşlev Çağrısı Grafik Görüntüleyicisi için varsayılan ayarlarda Performans Metriği Sentez ve Isı Haritası HİÇBİRİ olarak ayarlanmıştır.

4-3-2. Her işlev ve döngü için gecikme ve II değerlerini gözlemleyin.

İpucu: Bloklardaki metin görünmüyorsa, ilgilendiğiniz blokları yakınlaştırın. Yakınlaştırma düzeyi, metni kutuya sığdıracak kadar yüksek olduğunda, bu ayrıntılar görünecektir.

Isı haritası özelliği, ilgi çekici birkaç metriği vurgular:

o II (min, maks, ortalama)

o Gecikme (min, maks, ortalama)

o Durma Süresi Yüzdesi

Isı haritası, sorunlu modülleri kırmızıdan yeşile bir renk skalasıyla vurgulamak için renk kodlaması kullanır; burada kırmızı, metriğin yüksek bir değerini (kötü) (yani, en yüksek başlatma aralığı veya en yüksek gecikme) gösterirken yeşil (iyi) bir söz konusu metriğin düşük değeri. Ne kırmızı ne de yeşil olan renkler, en yüksek ve en düşük değerler arasındaki değer aralığını temsil eder.

4-3-3. Isı Haritası açılır okunu tıklayın ve ısı haritası seçeneklerini değiştirmeyi ve etkileri gözlemlemeyi deneyin.

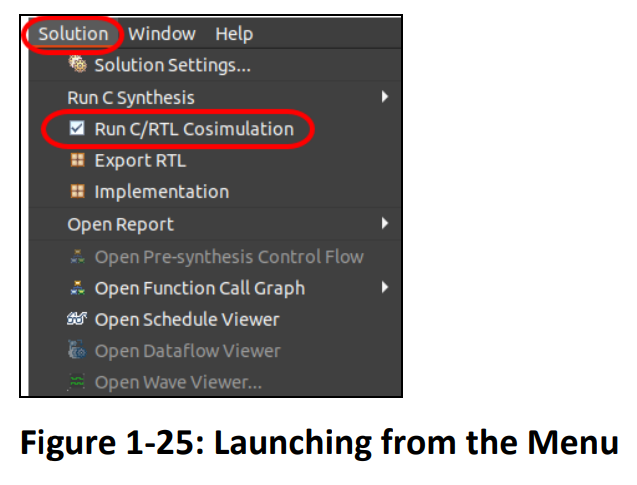
4-3-4. Tamamlandığında pencereyi kapatın.

**C/RTL Eş simülasyonu Gerçekleştirme Adım 5**

Artık C tasarımında üst düzey sentez gerçekleştirdiğinize göre, C test tezgahı kullanarak oluşturulan RTL üzerinde RTL ortak simülasyonu gerçekleştireceksiniz. Eş simülasyon (veya burada yaptığınız şeyle daha doğrusu simülasyon doğrulaması), yazılımın davranışını (C/C++ kodu) RTL kodunun davranışıyla karşılaştırır. Umut, aynı sonuçların üretilmesidir. Şimdi Verilog'u kullanarak C/RTL ortak simülasyonunu çalıştıracaksınız. Bu, VHDL kullanılarak da yapılabilir; bununla birlikte, zaman uğruna, yalnızca bir akış gösterilir. Bu sürecin sonunda her iki simülasyonun da aynı sonuçları verdiğini görmek istiyorsunuz.

5-1. Çözüm üzerinde ortak simülasyon çalıştırın. Bu, C/C++ ve RTL tasarımlarının simülasyon sonuçlarını karşılaştıracaktır.

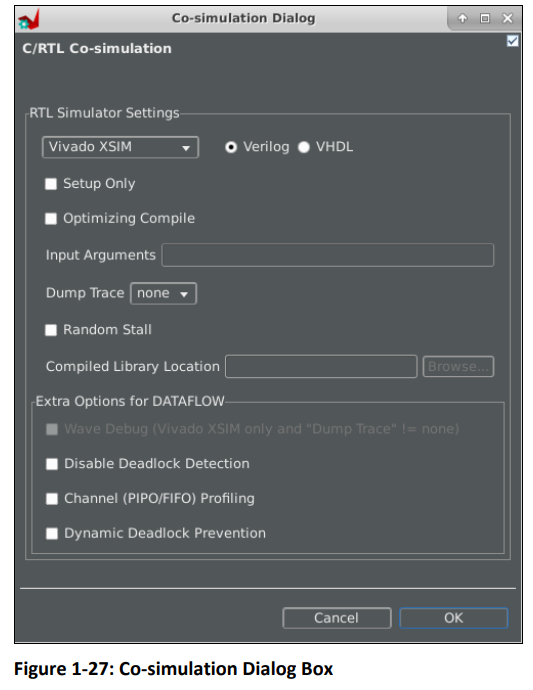
5-1-1. Eş simülasyonu başlatmak için Çözüm > C/RTL Kosimülasyonunu Çalıştır'ı seçin.



metin içeren bir resim

Açıklama otomatik olarak oluşturulduAlternatif olarak, Akış Gezgini'nden C/RTL Co-simulation'ı genişletin ve Run Cosimulation'a tıklayın.

Run C/RTL Co-simulation iletişim kutusu açılır.



İletişim kutusu aşağıdaki ayarları içerir:

o Simülatör: Vivado Design Suite'te desteklenen HDL simülatörlerinden birini seçin. Vivado simülatörü (Vivado XSIM) varsayılan simülatördür. Diğer simülatörler için lisans gerekebilir.

o Dil: Simülasyon için çıkış dili olarak Verilog veya VHDL'yi belirtin.

o Yalnızca Kurulum: Gerekli simülasyon dosyalarını oluşturur ancak simülasyonu çalıştırmaz. Yürütülebilir simülasyon dosyası daha sonra bir komut kabuğundan çalıştırılabilir.

o Derlemeyi Optimize Etme: Mümkünse, derlemesi genellikle daha uzun süren gelişmiş optimizasyon yoluyla çalışma zamanı performansını iyileştirin.

o Girdi Bağımsız Değişkenleri: C test tezgahı için ek komut satırı bağımsız değişkenlerini belirtin.

o Dump Trace: Simülasyon yürütüldüğünde mevcut çözümün sim/Verilog veya sim/VHDL dizinine yazılan izleme dosyası çıktısının seviyesini belirtir. Seçenekler şunları içerir:

▪ tümü: İzleme dosyasına kaydedilen tüm bağlantı noktası ve sinyal dalga biçimi verilerinin çıktısını alın.

▪ bağlantı noktası: Yalnızca üst düzey bağlantı noktaları için çıkış dalga biçimi izleme verileri.

▪ yok: İzleme verilerinin çıktısını almayın.

o Rastgele Durma: Her veri aktarımı için rastgele bir duraklama uygular.

5-1-2. Varsayılan seçenekleri seçin (yani hiçbir şeyi değiştirmeyin).

5-1-3. Tamam'ı tıklayın.

Simülasyon günlüğü Konsol bölmesinde görüntülenecektir.

Not: Konsol bölmesi, test tezgahı C kodu tarafından oluşturulan "Sonuçlar İyi" mesajını gösterir.

5-2. Kosimülasyon raporunu görüntüleyin.

Vitis HLS aracı tarafından oluşturulan bilgiler, her ikisi de burada açıklanan iki yerde bulunabilir.

Birincisi, yalnızca simüle edilen kodun ürettiği çıktıyı değil, aynı zamanda tüm simülasyon motoru mesajlarını da bildiren Konsol penceresidir.

Simülasyon günlüğü, yalnızca birkaç simülasyon motoru mesajı ve simüle edilmiş kod çıktısı sağlar.

5-2-1. Bir kerede daha fazla mesaj görebilmeniz için ekranı büyütmek için Konsol sekmesine çift tıklayın.

Kosimülasyon tarafından üretilen tüm çıktıları görüntülemek için kaydırmanız gerekebilir.

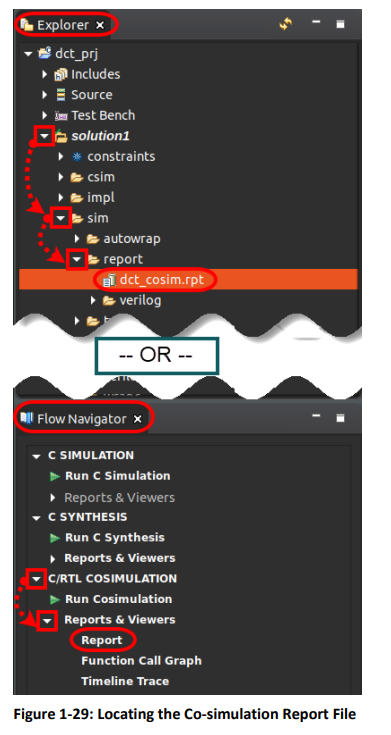
metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

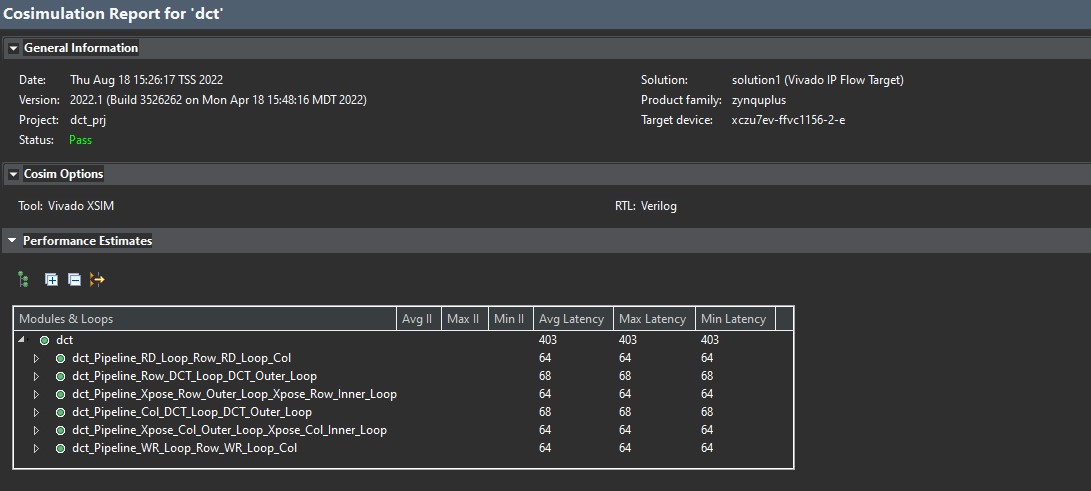
Daha sonra ziyaret edeceğiniz günlük dosyası (simülasyon raporu), yalnızca birkaç simülasyon motoru mesajı ve simüle edilmiş kod çıktısı sağlar. Tipik olarak, bu, simülasyon tamamlandıktan sonra açılır.

5-2-2. Gezgin bölmesini kullanarak dct\_prj > solution1 > sim > rapor öğesini genişletin.

5-2-3. Düzenleyici bölmesinde açmak için günlük dosyasına çift tıklayın.



Alternatif olarak rapor, C/RTL COSIMULATION > Raporlar ve Görüntüleyiciler > Rapor altındaki Akış Gezgini sekmesi kullanılarak bulunabilir.

Kosimülasyon Raporu, HTML formatında ana görüntüleme alanında görüntülenecektir. Ayrıca, C/RTL COSIMULATION bölümünün altındaki Raporlar ve Görüntüleyiciler bölümünü genişleterek Flow Navigator'dan kosimülasyon raporunu açabilirsiniz.

Kosimülasyon durumunu buradan hızlı bir şekilde doğrulayabilirsiniz. Bu işlemin tamamlanması birkaç dakika sürecektir. C/RTL kosimülasyonu tamamlandıktan sonra

Tamamlandığında, Cosimulation raporuna gecikme bilgileri de dahil olmak üzere Bilgi bölmesinden erişilebilir.

**Tasarımı IP Çekirdeğe Dönüştürme Adım 6**

Tasarımı başarıyla uyguladığınıza ve onayladığınıza göre, şimdi onu yeniden kullanılabilir IP'ye dönüştüreceksiniz.

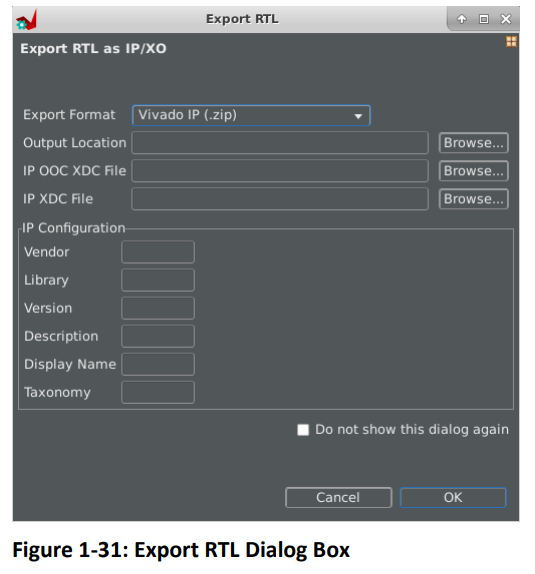
6-1. RTL tasarımını IP olarak dışa aktarın.

6-1-1. Dışa Aktarma Sihirbazını açmak için Çözüm > RTL'yi Dışa Aktar'ı seçin.

Alternatif olarak, Akış Gezgini > UYGULAMA'dan RTL'yi Dışa Aktar'a tıklayın.

6-1-2. Vivado aracında kullanılabilecek IP için uyumlu bir format tanımladığı için Dışa Aktarma Formatı açılır listesinden Vivado IP'sini (.zip) seçin.

Alternatif olarak, Vitis Kernel (.xo) seçildiğinde, uygulama hızlandırma geliştirme akışında Vitis derleyicisi tarafından bağlanmak için kullanılabilecek bir XO dosyası üretilir.



Aşağıdaki alanlara dikkat edin:

o Çıktı Konumu: Dışa aktarılan RTL tasarımı için yol ve dosya adını belirlemenizi sağlar. Varsayılan olarak, sıkıştırılmış tasarım, projenin çözüm dizini altına impl altında yerleştirilir.

o IP OOC XDC Dosyası: Bağlam dışı (OOC) sentez için RTL IP için kullanılacak bir XDC dosyasını belirtir.

o IP XDC Dosyası: Vivado IDE yeri ve rotası sırasında kullanım için bir XDC dosyası belirlemenizi sağlar.

o IP Yapılandırması: Satıcı, kitaplık, sürüm ve IP açıklaması gibi IP hakkında bilgiler sağlar.

6-1-3. Ayarları varsayılanlarında bırakın.

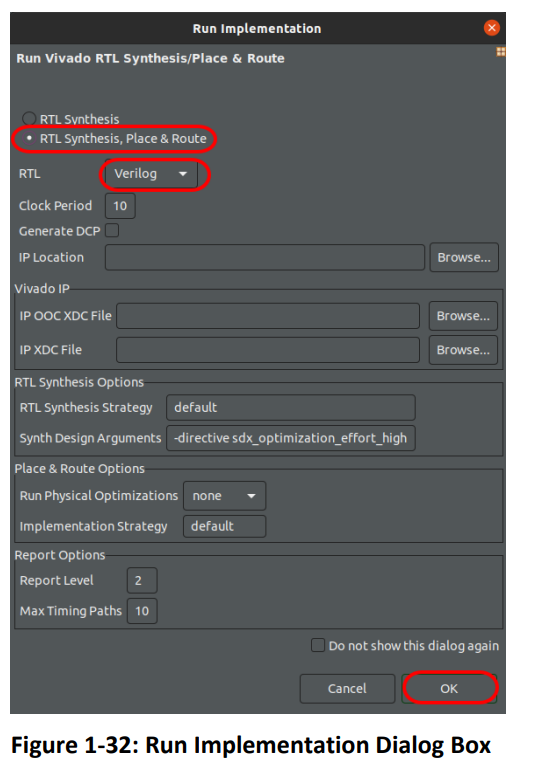
Vitis HLS aracı, ürettiği RTL tasarımı hakkında sağlayabileceği tahminler açısından sınırlıdır. Kaynak kullanımını ve nihai sonucun zamanlamasını yansıtabilir, ancak bunlar yalnızca tahminlerdir.

Vivado sentezini çalıştırarak ve oluşturulan RTL tasarımına yerleştirip yönlendirerek ve ardından Vivado Design Suite'te zamanlamayı ve kaynak kullanımını gözden geçirerek daha doğru değerler üretilir.

6-1-4. RTL'yi Dışa Aktar iletişim kutusunda Tamam'ı tıklayın. RTL başarıyla dışa aktarıldığında, uygulama adımına geçebilirsiniz.

6-2. Tasarımı uygulayın.

6-2-1. Uygulamayı Çalıştır iletişim kutusunu açmak için Çözüm > Uygulama'yı seçin. Alternatif olarak, Akış Gezgini'nden Uygulamayı Çalıştır'a tıklayın.



Bu iletişim kutusu, RTL sentezi veya RTL sentezi, yer ve rota çalıştırma seçeneklerini sunar. Yer ve rota seçeneklerinin altta görünmesi dışında, iletişim kutusu her iki seçimde de büyük ölçüde değişmez.

o RTL: Verilog veya VHDL formunda RTL üretir.

o Clock Period: Aktif çözüm tarafından varsayılan olarak tanımlanan saat periyodunu belirtir.

o Generate DCP: Sentezlenen veya uygulanan tasarım için bir DCP dosyası oluşturur.

o DCP Location: Bir önceki seçenek seçilirse, bu seçenek, oluşturulan DCP dosyasının yazılacağı konumu belirtir.

o IP Konumu: Oluşturulan IP'nin konumunu belirtir.

o IP OOC XDC Dosyası: Bağlam dışı (OOC) sentez için RTL IP için kullanılacak bir XDC dosyasını belirtir.

o IP XDC Dosyası: Vivado yeri ve rotası sırasında kullanım için bir XDC dosyası belirlemenizi sağlar.

o RTL Sentez Stratejisi: Sentez çalışmasında kullanılacak stratejiyi belirtir.

o Synth Design Argümanları: synth\_design komutu için seçenekleri belirtir.

o Rapor Düzeyi: Sentez veya uygulama sırasında oluşturulan rapor düzeyini tanımlar.

o Maks Zamanlama Yolları: Zamanlama Özeti raporundan çıkarılacak zamanlama yollarının sayısını belirtir. En kötü durum yolları, belirtilen değer tarafından tanımlandığı şekilde döndürülür.

o Fiziksel Optimizasyonları Çalıştır: Çalıştırılacak fiziksel optimizasyonu belirtir. Seçenekler şunlardır: hiçbiri, yer, rota ve tümü.

o Uygulama Stratejisi: Uygulama çalıştırmasında kullanılacak stratejiyi belirtir.

6-2-2. RTL Sentezi, Yerleştir ve Yönlendir seçeneğini seçin.

6-2-3. Değerlendirilecek RTL olarak Verilog'un seçildiğinden emin olun.

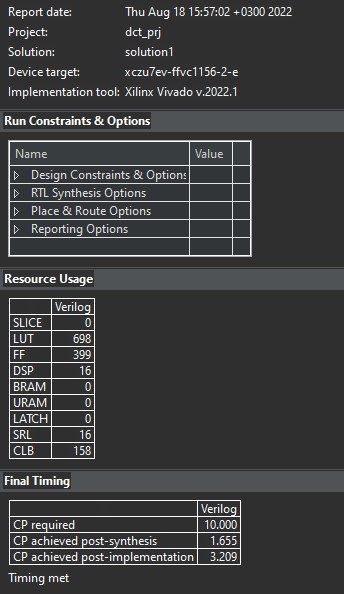
Bu seçenekler, oluşturulan IP üzerinde Vivado RTL sentezi ve uygulamasının gerçekleştirilmesini sağlayacaktır. Uygulama, RTL'nin tahmini zamanlama ve alan hedeflerini karşılayacağını ve bu sonuçların ihraç edilen paketin bir parçası olarak dahil edilmediğini değerlendirmek ve güven sağlamak için yürütülür.

6-2-4. Uygulamayı Çalıştır iletişim kutusunda Tamam'a tıklayın. Konsol sekmesinden ilerlemeyi gözlemleyebilirsiniz. Bu işlemin tamamlanması 4-5 dakika sürebilir. Uygulama tamamlandığında, Bilgi bölmesinde Uygulama raporu açılır.

Son Zamanlama bölümündeki rapordaki "Zamanlama karşılandı" mesajını gözlemleyin. Bu, uygulanan tasarımın son zamanlamasına ulaşıldığını gösterir.

metin içeren bir resim

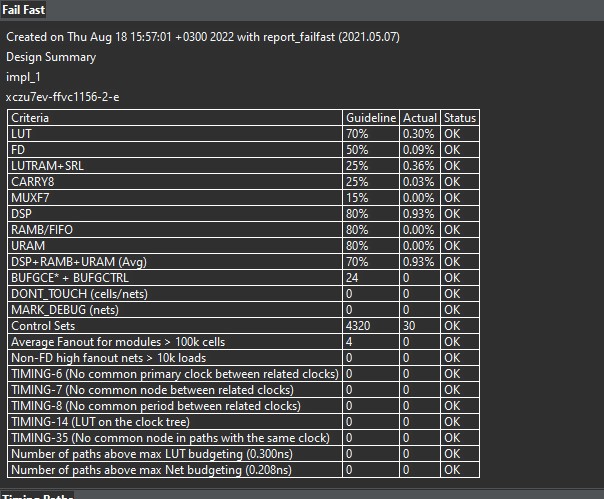
Açıklama otomatik olarak oluşturuldu



6-2-5. Uygulama raporunun Hızlı Başarısız bölümüne gidin.

Vivado Design Suite'in sunduğu Fail Fast raporları, aracın karşılaştığı belirli sorunlarla ilgili araştırmanıza rehberlik edebilir. Fail Fast raporunda, uygulamayı ve kapanış zamanını iyileştirmek için İNCELEME durumundaki her şeye bakmalısınız.

6-2-6. Uygulama raporunun Zamanlama Yolları bölümüne gidin.

Zamanlama Yolları raporları, tasarım için en kötü bolluğa neden olan zamanlama açısından kritik yolları gösterir. Varsayılan olarak, araç en kötü 10 negatif bolluk yolunu gösterecektir.Tablodaki her yol, bir flip-flop arasındaki kombinasyon yolunu gösteren ayrıntılı bilgilere sahiptir. Bu uzun kombinasyon yollarını kırmak, zamanlama sorunlarını çözmek için gerekli olacaktır. 

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Not: Değerleriniz sisteminize bağlı olarak biraz değişebilir.

6-2-7. Vitis HLS aracını kapatmak için Dosya > Çıkış'ı seçin

Bazı sistemler (özellikle VM'ler) bellek kısıtlı olabilir. Çalışma alanının kaldırılması, disk alanının bir kısmını boşaltarak diğer laboratuvarların gerçekleştirilmesine olanak tanır.

Az önce çalıştırdığınız laboratuvarı içeren dizini grafik arabirimi veya komut satırı arabirimini kullanarak silebilirsiniz. Her iki mekanizmayı da seçebilirsiniz. Her iki işlem de $TRAINING\_PATH/hls\_tool\_flow dizinindeki tüm dosyaları tekrar tekrar siler.

6-3. [İsteğe bağlı] [Yalnızca yerel sanal makineler için—CloudShare için değil] Dosya sistemini temizleyin.

GUI'yi kullanma:

6-3-1. Grafik tarayıcıyı kullanarak (Windows: <Windows> tuşu + <E>'ye basın; Linux: <Ctrl + N>'ye basın), $TRAINING\_PATH/hls\_tool\_flow.6-3-2'ye gidin. hls\_tool\_flow'u seçin.

6-3-3. <Sil>'e basın.

-- VEYA --

Komut satırını kullanarak:

6-3-4. Bir terminal penceresi açın (Windows: <Windows> tuşu + <R>'ye basın, ardından cmd girin;

Linux: <Ctrl + Alt + T> tuşlarına basın).

6-3-5. Çalışma alanının içeriğini silmek için aşağıdaki komutu girin: [Windows kullanıcıları]: rd /s /q $TRAINING\_PATH/hls\_tool\_flow

[Linux kullanıcıları]: rm -rf $TRAINING\_PATH/hls\_tool\_flow

**Özet**

Simülasyon, sentezleme, ortak simülasyon ve tasarımı bir IP olarak dışa aktarma dahil olmak üzere, temel geliştirme akışını izleyerek bir C programından bir donanım IP'si oluşturdunuz. Oluşturulan raporlardan bazılarını incelediniz ve tasarımın nasıl uygulandığını belirlediniz. Diğer laboratuvarlar burada öğrendiklerinizi genişletecektir.