İSTANBUL TEKNİK ÜNİVERSİTESİ
ELEKTRİK-ELEKTRONİK FAKÜLTESİ

H.264/AVC İÇERİSİNDE CABAC ENTROPİ KODLAMA BLOĞUNUN GERÇEKLENMESİ

LİSANS BİTİRME TASARIM PROJESİ

Fatih Enes DOĞAN

Yiğit Bektaş GÜRSOY

Uygundur
Berna Örs Yalçın
06.06.2023

ELEKTRONİK VE HABERLEŞME MÜHENDİSLİĞİ BÖLÜMÜ

MAYIS 2023

İSTANBUL TEKNİK ÜNİVERSİTESİ
ELEKTRİK-ELEKTRONİK FAKÜLTESİ

H.264/AVC İÇERİSİNDE CABAC ENTROPİ KODLAMA BLOĞUNUN GERÇEKLENMESİ

LİSANS BİTİRME TASARIM PROJESİ

Fatih Enes DOĞAN
040190745

Yiğit Bektaş GÜRSOY
040180063

Öğrenci Adı SOYADI
(Öğrenci No)

Proje Danışmanı: Prof. Dr. Sıddıka Berna Örs YALÇIN

ELEKTRONİK VE HABERLEŞME MÜHENDİSLİĞİ BÖLÜMÜ

MAYIS, 2023

İTÜ, Elektronik ve Haberleşme Mühendisliği Bölümü'nün ilgili Bitirme Tasarım Projesi yönergesine uygun olarak tamamen kendi çalışmamız sonucu hazırladığımız "H.264/AVC İÇERİSİNDE CABAC ENTROPİ KODLAMA BLOĞUNUN GERÇEKLENMESİ" başlıklı Bitirme Tasarım Projesi'ni sunmaktayız. Bu çalışmayı intihal olmaksızın hazırladığımızı taahhüt eder; intihal olması durumunda bitirme tasarım projesinin başarısız sayılacağını kabul ederiz.

Fatih Enes DOĞAN
040190745

Yiğit Bektaş GÜRSOY
040180063

ÖNSÖZ

Öncelikle bu projenin ortaya çıkmasındaki katkılarından ve yardımlarından ötürü Prof. Dr. Sıddıka Berna Örs YALÇIN ve yine projemizin her aşamasında bize yol gösteren Adem GÖLCÜK ve YONGATEK firmasına içtenlikle teşekkürlerimizi sunarız.

Hayatımızın her anında bizlerle olan ve bizlere destek veren ailelerimize ve sevdiklerimize teşekkür ederiz.

Mayıs 2023

Fatih Enes DOĞAN
Yiğit Bektaş GÜRSOY

İÇİNDEKİLER

Sayfa

KISALTMALAR

AVC : Advanced Video Coding
CABAC : Context-Adaptive Binary Arithmetic Coding
CAVLC : Context-Adaptive Variable-Length Coding
ctxIdxInc : Context Index Increment
DCT : Discrete Cosine Transform
FF : Flip Flop
FL : Fixed-Length
I/O : Input/Output
ITU : International Telecommunications Union
LUT : Look Up Table
MPS : Most Probable Symbol
PSNR : Peak signal-to-noise ratio
RTL : Register-transfer level
SE : Syntax Element
SSIM : Structural Similarity Index Measure
TU : Truncated Unary
U : Unary
UHD : Ultra High Definition

TABLO LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

H.264/AVC İÇERİSİNDE CABAC ENTROPİ KODLAMA BLOĞUNUN GERÇEKLENMESİ

ÖZET

Son yıllardaki teknolojideki ilerleyişlerle birlikte veri boyutlarında çarpıcı bir artış görülmektedir. Teknolojinin bu ilerleyişi, ilgili video kayıt cihazlarının kalitesinde de önemli rol oynamaktadır. Günümüzdeki video kayıt cihazlarıyla birlikte fotoğraf ve video boyutları da çarpıcı artışlar göstermektedir. Büyük boyutlu video ve fotoğraf dosyalarının depolanması ve iletilmesi ise, özellikle internet üzerindeki bant genişliği ve sunucu kapasiteleri gibi faktörler göz önünde bulundurulduğunda, çeşitli zorluklar ve problemlere yol açmaktadır. Bu nedenle, video sıkıştırma yöntemleri ve algoritmaları geliştirilmekte ve kullanılmaktadır. Video sıkıştırma yöntemleri, genellikle yüksek kaliteli videoların daha küçük boyutlarda saklanabilmesi ve hızlı bir şekilde iletilmesi amacıyla geliştirilmiştir. Bu sebeplerden ötürü çeşitli video sıkıştırma yöntemleri günümüzde büyük önem arz etmektedir. Günümüzde en çok kullanılan codeclerinden birisi olan H.264 sıkıştırma standardı, video sıkıştırma standardları arasında önde gelen bir standarttır. Bu sıkıştırma standardında kullanılan sıkıştırma algoritmasının ismi CABAC(Context-Adaptive Binary Arithmetic Coding)'dır. Bu sıkıştırma algoritması verilerin daha fazla sıkıştırılmasına imkan tanımaktadır CABAC, diğer sıkıştırma algoritmalarından farklı olarak verileri işlemek için istatiksel modellemeyi kullanır ve bu sayede yüksek kaliteli video yayınlarında birçok avantaj sağlamaktadır. CABAC sıkıştırma algoritması, aritmetik kodlama yöntemini kullanarak sembollerin olasılıklarını hesaplar. Bağlam modeline (context model) dayandırılan bu method sayesinde sıkıştırma işleminde daha yüksek bir verimlilik sağlar, böylece veri kaybını minimize eder. H.264/AVC standardında kullanılan CABAC, düşük bant genişliği ve depolama alanı kullanır, bunun sayesinde daha yüksek kaliteli video akışları sağlayabilir. Bunlara ek olarak, CABAC sayesinde ilerleyen sıkıştırma performansı, internet üzerinden yayınlanan video içeriğinin artmasıyla birlikte, video akışı hizmetleri için önemli bir konudur. CABAC, veri sıkıştırma işlemindeki yenilikçi yaklaşımı ve video sıkıştırma dünyasına getirdiği katkılarla önemli bir algoritmadır. Bu proje CABAC sıkıştırma algoritmasının literatürdeki yerinin araştırılmasını, Uluslararası Telekomünikasyon Birliği (ITU) tarafından yayınlanmış olan "Advanced Video Coding for Generic Audiovisual Services" başlıklı H.264 codec'inin tekniğini içeren standardın ve aynı birlik tarafından yayınlanan ve C dilinde yazılmış olan JM referans yazılımının incelenip anlaşılmasını, bu standardın MATLAB üzerinde tasarlanmasını ve doğrulanmasını ardından bu doğrulanan kodların RTL seviyesine geçirilmesini içermektedir. ITU tarafından yayınlanan standartta belirtilen algoritmalar ve fonksiyonlar MATLAB üzerinde tasarlandı ve gerekli bloklar üst seviye tasarımda birleştirildi. Daha sonra bu tasarımın doğrulanabilmesi için JM referans yazılımında birtakım düzenlemeler yapılarak .yuv uzantılı bir videonun kodlanması sırasında CABAC ile ilgili girdi değerlerini bir dosyaya yazdırabilecek hale getirildi. Dosyaya yazılmış bu girdiler MATLABda okunarak tasarımımıza verildi ve MATLAB çıktısı ile JM referans yazılımının çalışması sonucunda çıkan .264 uzantılı sıkıştırılmış video karşılaştırıldı. Yaklaşık 3500 frame üzerinde yapılan bu doğrulama aşamasında tasarımımız ile JM referans yazılımı tamamen eşleştiği gözlemlendi. Doğrulamanın ardından MATLAB

üzerindeki tasarım Vivado programı kullanılarak VHDL dilinde RTL seviyesinde kodlaması yapılmıştır.

REALIZING THE CABAC ENTROPY CODING BLOCK IN H.264/AVC

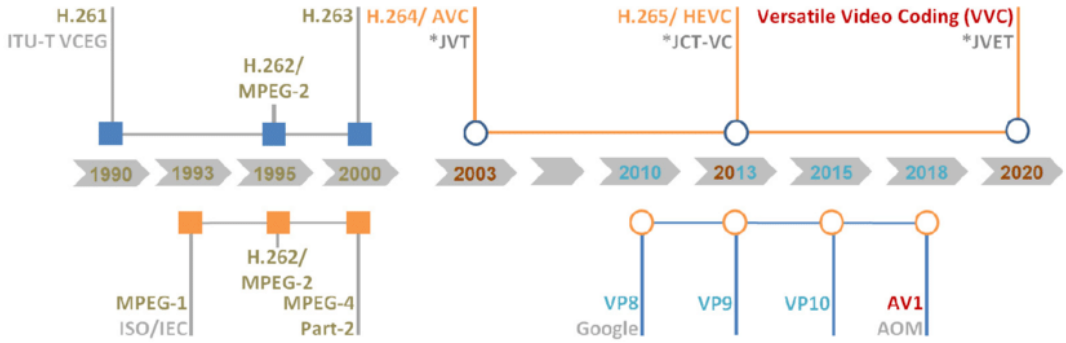
SUMMARY

With the advances in technology in recent years, there has been a dramatic increase in data sizes. This advancement of technology also plays an important role in the quality of related video recorders. With today's video recorders, photo and video sizes also increase dramatically. The storage and transmission of large video and photo files cause various difficulties and problems, especially when factors such as bandwidth and server capacities on the Internet are taken into account. Therefore, video compression methods and algorithms are being developed and used. Video compression methods are generally developed for the purpose of storing high quality videos in smaller sizes and transmitting them quickly. For these reasons, various video compression methods are of great importance today. H.264 compression standard, one of the most used codecs today, is a leading standard among video compression standards. The name of the compression algorithm used in this compression standard is CABAC (Context-Adaptive Binary Arithmetic Coding). This compression algorithm allows further compression of data. Unlike other compression algorithms, CABAC uses statistical modeling to process data and thus provides many advantages in high-quality video broadcasts. The CABAC compression algorithm calculates the probabilities of symbols using the arithmetic encoding method. This method, which is based on the context model, provides a higher efficiency in the compression process, thus minimizing data loss. Used in the H.264/AVC standard, CABAC uses low bandwidth and storage space, so it can provide higher quality video streams. In addition, improved compression performance thanks to CABAC is a key issue for video streaming services, with the rise of video content streamed over the internet. CABAC is an important algorithm with its innovative approach in data compression and its contributions to the world of video compression.

This project aims to investigate the place of the CABAC compression algorithm in the literature, the standard containing the technique of the H.264 codec titled "Advanced Video Coding for Generic Audiovisual Services" published by the International Telecommunication Union (ITU), and the JM reference software published by the same union and written in C language. It includes examining and understanding this standard, designing and verifying this standard on MATLAB, and then passing these verified codes to the RTL level. Algorithms and functions specified in the standard published by ITU were designed on MATLAB and the necessary blocks were combined in a high-level design. Later, in order to verify this design, some adjustments were made to the JM reference software, and it was made to be able to print CABAC-related input values to a file while encoding a .yuv video. These inputs written to the file were read in MATLAB and given to our design, and the MATLAB output and the compressed video with the .264 extension, which came as a result of the work of the JM reference software, were compared. It was observed that our design and JM reference software completely matched during this verification phase, which was carried out on approximately 3500 frames. After the verification, the design on MATLAB was coded at the RTL level in the VHDL language using the Vivado program.

1. GİRİŞ

Günümüzde teknolojinin birçok alanda ilerlemesinde sayesinde kayıt cihazları yüksek kalitede veri depolayabilmektedir. Bu cihazlar ile kaydedilen verilerin boyutları de giderek artmaktadır. Bu artışlar verilerin depolanmasında ve işlenmesinde sıkıntılara yol açmaktadır. Bu sorunlar sonucunda verileri depolayarak onların boyutunu küçülten ve daha fazla veri depolanmasına olanak sağlayan video standartları geliştirilmiştir. Video sıkıştırma standartları getirdiği yeniliklerle veri iletimi, depolama ve video işleme süreçlerini mümkün kılmaktadır. Başlıca geliştirilen standartları H.264, H.265 ve H266 şeklinde sayabiliriz ve bu standartlar özellikle endüstrideki talepleri ve beklentileri karşılamak için geliştirilmiştir. Aşağıda video sıkıştırma standartlarının tarihsel gelişimine ait görsel belirtilmiştir.



Şekil 1.1 Video Sıkıştırma Standartlarının Tarihsel Gelişimi[3]

Bu standartların geliştirilmesi ve kullanılması, internet üzerinden gerçekleştirilen canlı yayınların veya yüksek çözünürlüklü videoların akışlarının kullanıcı deneyimini yüksek verimlilikte, olumsuz etkilemeden ve bant genişliğini en aza indirerek, sağlamıştır.

2. H264 STANDARDINA GENEL BAKIŞ

2.1 Genel

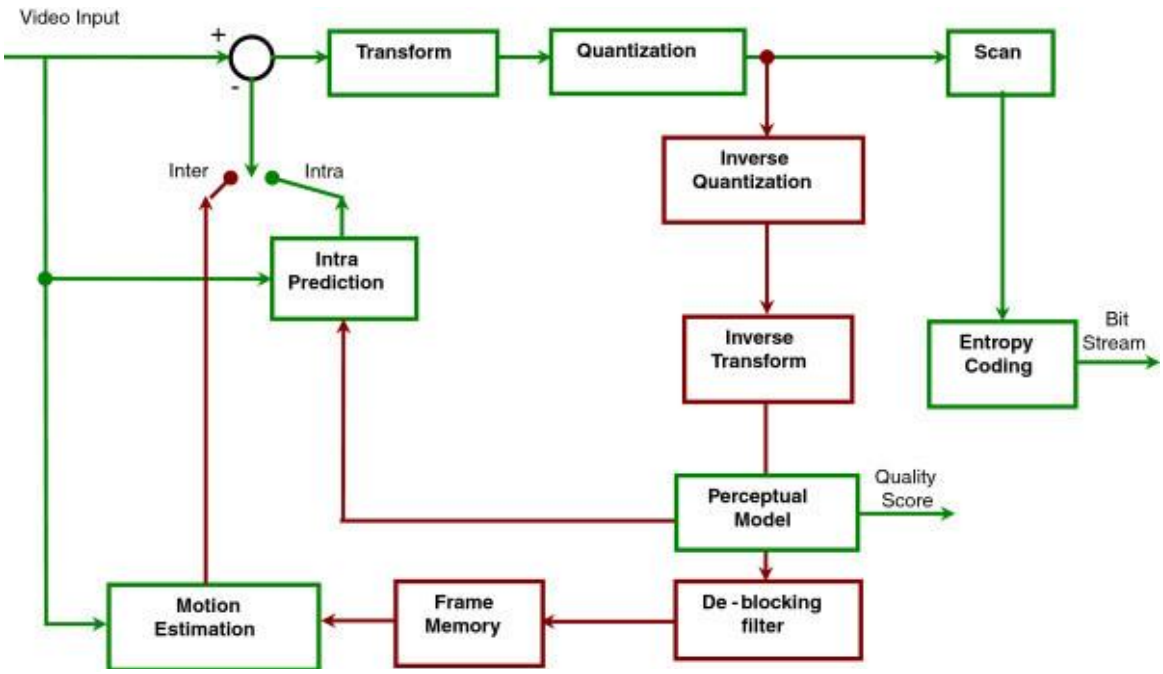
H.264 blok yönelimli ve hareket dengelemeli kodlamaya[7] bağlı görüntü sıkıştırma standardıdır ve MPEG-4 AVC sıkıştırma standardının eşdeğeri olarak görülür. H.264 standardı maksimum 8K UHD çözünürlüğü destekler.[8]

H.264 sıkıştırma standardının tasarlanma amacı önceki sıkıştırma standartlarından daha düşük bit hızlarında daha iyi bir video kalitesi sağlayabilmesidir. Bu özelliklerin sağlanması amacıyla H.264 geliştirilirken kullanılan yöntemlerden bazıları şunlardır: Azaltılmış karmaşıklıkta tamsayı ayrık kosinüs dönüşümü (tamsayı DCT),[9] değişken blok boyutlu segmentasyon ve çoklu resim tahmini... Belirlenmiş bir decoder en az bir profilin kodunu çözebilir ama bu decoder tüm profiller için uygun olmayabilir. H.264 sıkıştırma standartı, video kodlanması amacıyla algoritma belirlemez ama kodladığı verilerin türünü ve verilerin kodlarının nasıl çözüldüğünü açıklar. Bunun sebebi ise ilgili kaynak dosyalarını kodlarken tasarımcıların tercihlerine bırakılmasından dolayıdır ve bu uygulama çok sayıda kodlama şemalarının geliştirilmesine olanak sağlamıştır. Kayıplı sıkıştırmalarda genellikle H.264 standartı tercih edilir ama bu standart sayesinde kodlamanın tümünün kayıpsız olduğu ender durumları desteklemek de mümkündür.

H.264, Blu-ray disklerde en sık kullanılan video kodlama standartlarından birisidir ve ek olarak sık olarak kullanıldığı bazı internet video yayın platformları şunlardır: Netflix, Amazon Prime Video, YouTube, iTunes Store ,Adobe Flash Player, Microsoft Silverlight…

2.2 Bloklar

H.264 daha iyi anlamak için blok diyagramına göz atılabilir. Aşağıda H.264 sıkıştırma standartına ait bir fotoğraf belirtilmiştir. Birçok işlem bloğuna sahip olan bu standart, bloklarında yaptığı uygulamalar aşağıda belirtilmiştir.



Şekil 2.1 H.264 sıkıştırma formatının blok diyagramı[11]

- Video Input[1]: Bu blokta işlenecek ve kodlanacak sıkıştırılmamış video verisi bulunur.

- Transform[10]: H.264, ayrık kosinüs dönüşümüne (DCT) yaklaşan bir tamsayı dönüşümü kullanır. Bu blokta bu yöntemin kullanılmasının amacı, görüntüdeki uzamsal fazlalığı sıkıştırmanın daha kolay olduğu frekans alanına dönüştürmektir.

- Kuantizasyon[1]: Bu blok, dönüştürme katsayılarının kesinliğini azaltır. Amaç, insan görüntüleyenler tarafından fark edilme olasılığı en düşük olan verileri kaldırmaktır, bu da veri boyutunun küçültülmesine yardımcı olur.

- Scan[1]: Bu blok, nicelenmiş katsayıların 2 boyutlu dizisini 1 boyutlu diziye yeniden düzenler. Bu düzenlemeyi de genellikle daha büyük olma yatkınlığında olan düşük frekanslı katsayıları bir arada tutmak için bir zig-zag modelini takip eder.

- Inverse Quantization ve Inverse Transform[1]: Bu iki blok, decoder tarafındaki niceleme ve dönüştürme süreçlerini ters çevirir. Sıkıştırılmış veriler tekrardan video haline dönüştürülür.

- Perceptual Model ve Quality Score[12]: Bu süreçte dolaylı yoldan H.264 sıkıştırma formatının nasıl kodlama yaptığını gösterir. **V**ideo kalitesini

korumak amacıyla insan algısını referans alabilir. Kalite puanı, PSNR (Tepe Sinyal-Gürültü Oranı) veya SSIM (Yapısal Benzerlik İndeksi Ölçümü) gibi bir ölçüyü temsil edebilir.

- Entropy Coding[1]: Bu blok sıkıştırma algoritmalarının bulunduğu bloktur. Bu blok kayıpsız sıkıştırma teknikleri kullanılarak verilerin daha da sıkıştırılmasını sağlar. H.264 sıkıştırma standartı için 2: Context-Adaptive Variable-Length Coding (CAVLC) ve Context-Adaptive Binary Arithmetic Coding (CABAC).

- Bitstream**[1]**: Entropi kodlamasının çıktısı, sıkıştırılmış video verilerini temsil eden bir bit dizisi olan bir bit akışıdır.

- De-blocking filter**[1]:** Bu engelleme, videonun algılanan kalitesini iyileştirerek, kodlama işlemi tarafından oluşturulabilen engelleme yapılarını azaltır.

- Frame Memory[1]: Bu blokta önceden işlenmiş kareler bulunur ve bu kareler tahminde kullanılmak üzere depolanır.

- Motion Estimation**[1]:** Bu blokta hareket vektörleri üretilir. Bu hareket vektorleriyle birlikte geçerli frame ile önceki frameler arasındaki hareketi tahmin eder.

- Intra Prediction[1]: Bu blok, uzamsal fazlalığı azaltmak için aynı frame içinde bir makroblok öngörür.

- Intra/Inter[1]: Bu işlevler, H.264 sıkıştırma standardında iki farklı tahmin türüdür. Tahmin mevcut frame içinden yapıldığında intra prediction kullanılırken, tahmin farklı bir frameden yapıldığında inter prediction kullanılır.

2.3 Profiller

H.264, her biri farklı uygulamalara veya kullanım durumlarına hitap eden bir dizi yetenek tanımlayan birkaç profil içerir. Sıklıkla kullanılan profiller aşağıdaki tabloda belirtilmiş olup bazılarının da tanımlaması aşağıda yapılmıştır:

Feature	CBP	BP	XP	MP	ProHiP	HiP	Hi10P	Hi422P	Hi444PP
Bit depth (per sample)	8	8	8	8	8	8	8 to 10	8 to 10	8 to 14
Chroma formats	4:2:0	4:2:0	4:2:0	4:2:0	4:2:0	4:2:0	4:2:0	4:2:0/ 4:2:2	4:2:0/ 4:2:2/ 4:4:4
Flexible macroblock ordering (FMO)	No	Yes	Yes	No	No	No	No	No	No
Arbitrary slice ordering (ASO)	No	Yes	Yes	No	No	No	No	No	No
Redundant slices (RS)	No	Yes	Yes	No	No	No	No	No	No
Data Partitioning	No	No	Yes	No	No	No	No	No	No
SI and SP slices	No	No	Yes	No	No	No	No	No	No
Interlaced coding (PicAFF, MBAFF)	No	No	Yes	Yes	No	Yes	Yes	Yes	Yes
B slices	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
CABAC entropy coding	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes
4:0:0 (Monochrome)	No	No	No	No	Yes	Yes	Yes	Yes	Yes
8×8 vs. 4×4 transform adaptivity	No	No	No	No	Yes	Yes	Yes	Yes	Yes
Quantization scaling matrices	No	No	No	No	Yes	Yes	Yes	Yes	Yes
Separate C_B and C_R QP control	No	No	No	No	Yes	Yes	Yes	Yes	Yes
Separate color plane coding	No	No	No	No	No	No	No	No	Yes
Predictive lossless coding	No	No	No	No	No	No	No	No	Yes

Şekil 2.2 H.264 profilleri ve özellikleri

- Baseline Profile (BP)[1]: Sınırlı bilgi işlem kaynakları ve yetenekleri olan düşük maliyetli uygulamalar için tasarlanmıştır. Bu profil, B (çift yönlü tahmini) çerçeveleri desteklemez ve sınırlı sayıda özelliğe sahiptir. Genellikle video konferans ve mobil uygulamalarda kullanılır.

- Main Profile (MP)[1]: Bu profil, CABAC sıkıştırma algoritmasının çalışabildiği minimum profildir. Standart tanımlı video için Temel

Profilden üstün kalite sağlar. Standart tanımlı yayınlarda yaygın olarak kullanılmaktadır.

- Extended Profile (XP) [1]: Bu profil, video akışı için tasarlanmıştır. Baseline Profile'nin tüm özelliklerini içerir ayrıca video kalitesini ve sıkıştırma verimliliğini artıran B-frameleri gibi birkaç ek özellik de içerir.

- High Profile (HiP) [1]: Bu profilde CABAC algoritmasının verimliliği CAVLC algoritmasının verimliliğinde daha yüksektir.[1] Genellikle yüksek kalitedeki videolar, HDTV ve Blu-ray disk dahil olmak üzere, için kullanılan High Profil, Main Profilinin tüm özelliklerini içerir, bu özelliklerine ek olarak ise sıkıştırma verimliliğini daha da artırmak için birkaç özellik daha içerir.

- High 10 Profile (Hi10P) [1]: Bu profil, High Profil'in tüm özelliklerini içerir. Renk bileşeni başına 10 bit'e kadar destek ekler. Genellikle profesyonel uygulamalar için kullanılır.

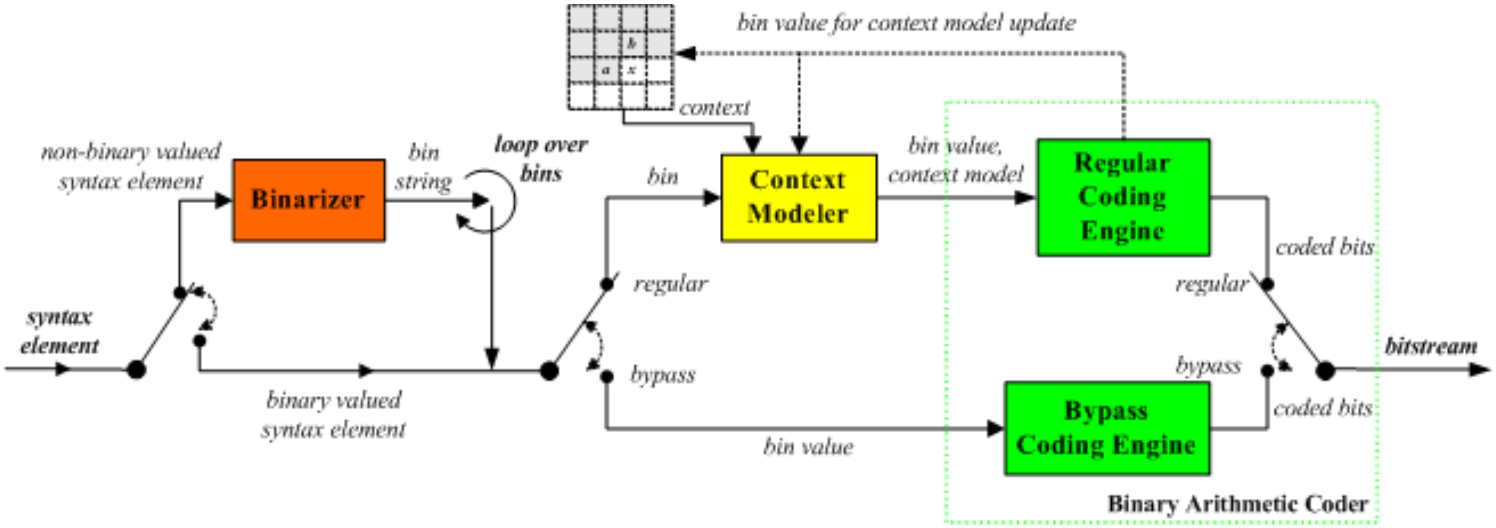
- High 4:2:2 Profile (Hi422P)[1]: Hi10P'nin tüm özelliklerini içeren ve 4:2:2'de chroma alt örneklemeyi destekleyen bu profil, profesyonel uygulamalar için tasarlanmıştır.

- High 4:4:4 Predictive Profile (Hi444PP) [1]: Bu profil, kayıpsız video kodlamasının yanı sıra renk bileşeni başına 14 bit'e kadar ve RGB renk alanını desteklemesi gibi özelliklere sahiptir. Bu tür gelişmiş özellikler gerektiren profesyonel uygulamalar için tasarlanmıştır.

3. CABAC GENEL BAKIŞ

CABAC (Context-Adaptive Binary Arithmetic Coding) video sıkıştırma algoritması genelde H.264/AVC ve H.265/HEVC standartları için kullanılır. CABAC, video sıkıştırma işleminin entropi kodlama kısmında yer alır. Veri akışını sıkıştırarak depolama veya iletim için gereken bit miktarını en aza indirir. CABAC sıkıştırma algoritması bitler üzerinden işlem yapan bir algoritmadır. Verileri daha verimli kodlamak amacıyla bu bitlerin olasılıklarını öğrenir ve bu olasılıkları kullanır . Bu

özellikler sayesinde daha küçük bir bit akışı daha az depolama alanı ve daha hızlı iletim süresi sunar[1]. Aşağıda CABAC sıkıştırma algoritmasının blok diyagramı belirtilmiştir.



Şekil 3.1 CABAC blok diyagramı[2]

CABAC sıkıştırma algoritmasının SE girişinden bitstream çıkışına gidene kadar belli aşamaları vardır[1]:

- Binarizer: CABAC, ikili aritmetik kodlamayı kullanır ve bu da yalnızca ikilik tabanda işlem yaptığı anlamına gelir. İkilik tabanda olmayan bir sembol aritmetik kodlamadan önce ikilik tabanda yazılır. Bu işlem bir veri sembolünü değişken uzunluklu bir koda dönüştürme işlemiyle benzerlik gösterir fakat ikili kod, iletimden önce aritmetik kodlayıcı tarafından ayrıca kodlanır.

- Context Modeler: Bir "bağlam modeli" ikilik tabanda temsil edilen bir veya birden fazla sembol için bir olasılık modelidir. Yakın zamanda kodlanmış veri sembollerinin istatistiklerine bağlı olarak mevcut modellerden seçilir. Bağlam modeli, her ikileştirmenin "1" veya "0" olma olasılığını saklar.

- Binary Arithmetic Coder: Bir aritmetik kodlayıcı, her ikilileştirmeyi seçilen olasılık modeline göre kodlar. Her ikilileştirme için muhtemel '0' ve '1' değerlerine karşılık gelen yalnızca iki alt aralık vardır.

- Bin value for context model update: Seçilen bağlam modeli, gerçek kodlanmış değere göre güncellenir. Örnek vermek gerekirse ikilileştirme değeri '1' ise, '1'lerin frekansı artırılır.

3.1 Bloklar

3.1.1 Binarizer

3.1.1.1 Unary binarization (U)

Binarization işleminden geçecek SE'nin değeri kadar 1 ve en son bit olarak da 0'dan oluşan bitstream çıktısı veren yöntemdir. Bitstream uzunluğu SE'nin değeri+1 olarak hesaplanır.

SE Değeri	Bitstream
0	0
1	10
2	110
3	1110
4	11110
5	111110
6	1111110
7	11111110

Tablo 3.1 : SE Değerlerine Göre Unary Binarization Çıktısı

3.1.1.2 Truncated unary binarization (TU)

Unary Binarization yöntemine oldukça benzeyen bir yöntemdir. SE değeri yanında bir cMax değeri ile çalışır. cMax değeri ikilileştirilebilecek en büyük SE değerini ifade eder[kaynak ITU standart]. cMax değerinden küçük SE değerleri için Unary yöntemi ile aynı şekilde çalışmaktadır ve bitstreamin uzunluğu SE değeri + 1 olarak hesaplanır. SE değeri cMax

değerine eşit olduğunda ise Unary yöntemindeki son bit olan 0 biti 1 olur ve bitstreamin uzunluğu SE değerine eşit olur.

SE Değeri	Bitstream
0	0
1	10
2	110
3	1110
4	11110
5	111110
6	1111110
7	11111111

Tablo 3.2 : cMax = 7 iken SE Değerlerine Göre Truncated Unary Binarization Çıktısı

3.1.1.3 Fixed-Length binarization (FL)

TU binarization yöntemi gibi SE değerinin yanında cMax değeri ile çalışmaktadır. cMax değeri yine ikilileştirilebilecek en büyük SE değerini ifade etmektedir. Bitstreamin uzunluğu her cMax için sabit olacaktır ve bu değer ceil($\log_2$(cMax+1)) yöntemi ile hesaplanır.

SE Değeri	Bitstream
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Tablo 3.3 : cMax = 7 iken SE Değerlerine Göre Fixed-Length Binarization Çıktısı

3.1.1.4 Birleştirilmiş unary/k mertebeli Exp-Golomb binarization(UEGk)

Bu yöntemin girdileri SE değeri, uCoff, signedValFlag, k isimli değişkenlerdir. Önek ve sonek olarak iki kısımdan oluşur. Önek kısmında kodlanacak değer min(uCoff, abs(SE_değer)), cMax ise uCoff olacak

şekilde TU Binarization yöntemi kullanılır. Eğer ki şu iki şarttan birisi sağlanırsa binarization çıkışı yalnızca önek kısmından oluşur;

1) signedValFlag = 0 ve önek uCoff uzunluklu ve sadece "1" içeren bir bitstream değil.
2) signedValFlag = 1 ve önek "0" değil.

Diğer koşullarda binarization çıkışında önek kısmına ek olarak sonek kısmı eklenir. Sonek kısmı k. Derece Exp-Golomb algoritması ile hesaplanır ve pseudo kodu şu şekildedir.

```
if( Abs( synElVal ) >= uCoff ) {
    sufS = Abs( synElVal ) − uCoff
    stopLoop = 0
    do {
        if( sufS >= ( 1 << k ) ) {
            put( 1 )
            sufS = sufS − ( 1<<k )
            k++
        } else {
            put( 0 )
            while( k− − )
                put( ( sufS >> k ) & 1 )
            stopLoop = 1
        }
    } while( !stopLoop )
}
if( signedValFlag && synElVal != 0)
    if( synElVal > 0 )
        put( 0 )
    else
        put( 1 )
```

Şekil 3.2 : UEGk Binarization için sonek kısmının pseudo kodu.

SE'in Mutlak Değer	TU ile Kodlanan Önek	EGk ile Kodlanan Sonek
1	0	-
2	10	-
3	110	-
4	1110	-
5	11110	-
6	111110	-
7	1111110	-
…	…	…
…	…	…
13	1111111111110	-
14	11111111111110	-
15	11111111111111	0
16	11111111111111	100
17	11111111111111	101
18	11111111111111	11000
19	11111111111111	11001
20	11111111111111	11010
21	11111111111111	1110000
…	…	…

Tablo 3.4[7]: coeff abs level minus1 SE'i için UEG0, uCoff = 14 binarization tablosu

3.1.1.5 Coded_block_pattern için binarization yöntemi

Coded Block Pattern SE'inin binarization işlemi önek ve sonek olmak üzere iki kısımdan oluşur. Önek kısmı CodedBlockPatternLuma isimli değişkeninin, cMax = 15 ile Fixed-Length binarizationdan geçmesi ile oluşmaktadır. Eğer ChromaArrayType isimli değişken 0 veya 3 ise SE'in binarization'ı yalnızca önekten oluşur. Aksi durumlarda CodedBlockPatternChroma değişkeninin cMax = 2 ile Truncated Unary Binarizationdan geçmesi ile sonek oluşur. CodedBlockPatternLuma ve CodedBlockPatternChroma değişkenleri ise aşağıdaki şekilde hesaplanır:

- CodedBlockPatternLuma = Coded_Block_Pattern % 16
- CodedBlockPatternChroma = Coded_Block_Pattern / 16

3.1.1.6 Mb_qp_delta için binarization yöntemi

Aslında mb_qp_delta SE'i için yalnızca Unary Binarization yöntemi kullanılmaktadır. Bu SE'in binarization sürecini özel yöntemlere alma sebebimiz ise mb_qp_delta SE'inin değerinin direkt olarak bu binarization yöntemine girmek yerine öncelikle başka bir değere eşlendikten sonra yöntemden geçmesidir. Aşağıdaki tabloda SE değeri ve eşlenecek değerler görülmektedir. Bu eşleme sonrasında yalnızca Unary Binarization yönteminden geçer ve çıkışta sonucumuzu görürüz.

SE Değeri	Eşlenecek Değer
0	0
1	1
-1	2
2	3
-2	4
3	5
-3	6
k pozitif	2*k - 1
k negatif	2*abs(k)

Tablo 3.5 : mb_qp_delta değeri ve eşlenecek değer.

3.1.1.7 I slice mb_type için binarization

I Slice mb_type SE'i için binarization tablo kullanılarak gerçekleştirilir. Aşağıdaki tabloda SE değerleri için binarization çıktısını görmekteyiz.

mb_type değeri	Binarization Çıktısı
0	0
1	100000
2	100001
3	100010
4	100011
5	1001000
6	1001001
7	1001010
8	1001011
9	1001100
10	1001101
11	1001110
12	1001111
13	101000
14	101001
15	101010
16	101011
17	1011000
18	1011001
19	1011010
20	1011011
21	1011100
22	1011101
23	1011110
24	1011111
25	11

Tablo 3.6: I Slice mb_type için Binarization Tablosu

3.1.1.8 P slice mb_type için binarization

P Slice mb_type SE'i için binarization yine bir tablo yardımı ile gerçekleştirilir. Eğer mb_type değeri 5'ten büyükse binarization önek ve sonek olarak iki kısımda hesaplanır. 5'ten küçük değerler için yalnızca önek vardır ve aşağıdaki tablodaki değeri alır. P Slice için mb_type değeri 4 olamaz. Mb_type değerinin beşten büyük olması durumunda önek yalnızca "1" olur ve mb_type değerinden

5 çıkartılarak “I Slice mb_type binarization” tablosuna bakılır ve o değer sonek olur. Örneğin mb_type değerimiz 13 olsun. Aşağıdaki tabloya göre önek “1” olacaktır. Sonek için ise gereken değer 13 - 5 = 8 ile hesaplanır. “I Slice mb_type binarization” tablosuna göre 8 değerine karşılık binarization çıktısı ise “1001011” olacaktır. Bu durumda 13 için P Slice mb_type binarization çıktısı “11001011” olacaktır.

mb_type değeri	Binarization Çıktısı
0	000
1	011
2	010
3	001
4	na
5’ten 30’a (yalnızca önek)	1

Tablo 3.7: P Slice mb_type Binarization Tablosu

3.1.1.9 B slice mb_type için binarization

B Slice mb_type da P Slice mb_type’a benzer bir binarization sürecinden geçer. 23’ten küçük değerler için çıktı, yalnızca aşağıdaki tabloya göre önekten oluşur. 23’ten büyük bir değerin kodlanması istendiğinde ise mb_type değerinden 23 çıkarılır ve aynı P Slice mb_type’ta olduğu gibi yeni değer ile “I Slice mb_type Binarization” tablosunda bakılır. Örneğin 30 değeri binarizationdan geçirilmek istensin; önek kısmı tablodan görüleceği üzere “111101” olacaktır. Sonek kısmı için kodlanacak değer ise 30 – 23 = 7 ile hesaplanır ve gerekli tablodan binarization çıktısına bakılır. Bu değer “1001010” olacaktır. Yani 30 değerinin binarization çıktısı “1111011001010” olur.

mb_type değeri	Binarization Çıktısı
0	0
1	100
2	101
3	110000
4	110001
5	110010
6	110011
7	110100
8	110101
9	110110
10	110111
11	111110
12	1110000
13	1110001
14	1110010
15	1110011
16	1110100
17	1110101
18	1110110
19	1110111
20	1111000
21	1111001
22	111111
23'ten 48'e (Yalnızca Önek)	111101

Tablo 3.8: B Slice mb_type Binarization Tablosu

3.1.1.10 P slice sub_mb_type için binarization

P Slice sub_mb_type SE'i için binarization mb_type için olduğu gibi tablo yardımı ile hesaplanır. Sadece tek kısımdan oluşur. Sub_mb_type değeri için binarization çıkışı aşağıdaki tablodan görülebilir.

sub_mb_type değeri	Binarization Çıktısı
0	1
1	00
2	011
3	010

Tablo 3.9: P Slice sub_mb_type Binarization Tablosu

3.1.1.11 B slice sub_mb_type için binarization

B Slice sub_mb_type SE'i için binarization mb_type için olduğu gibi tablo yardımı ile hesaplanır. Sadece tek
kısımdan oluşur. Sub_mb_type değeri için binarization çıkışı aşağıdaki tablodan görülebilir.

sub_mb_type değeri	Binarization Çıktısı
0	0
1	100
2	101
3	11000
4	11001
5	11010
6	11011
7	111000
8	111001
9	111010
10	111011
11	11110
12	11111

Tablo 3.10: B Slice sub_mb_type Binarization Tablosu

3.1.2 Context modeler

Context model bloğu, kayıpsız veri sıkıştırma için yaygın olarak kullanılan bir entropi kodlama tekniği olan CABAC sıkıştırma algoritmasının önemli bileşenlerinden birisidir. Context model bloğu, sıkıştırılan verilerdeki sembollerin olasılık dağılımının belirlenmesinde önemli bir rol oynamaktadır. Aşağıda context modeler bloğunun içerisinde gerçekleşen bazı önemli aşamalardan bahsedilmiştir:

- Her SE, o SE'nin bölgesel istatistiksel özelliklerini temsil eden belirli bir context atanır. Context model bloğu, her bir context için sembollerin olasılık dağılımının modellenmesinden ve tahmin edilmesinden sorumludur.
- Context model bloğu, bir dizi binary context modelinden oluşur. Context model, temelde aynı contextdeki önceki sembollere dayalı olarak bir ikili sembolün (0 veya 1) oluşma olasılığını yakalayan istatistiksel bir modeldir.

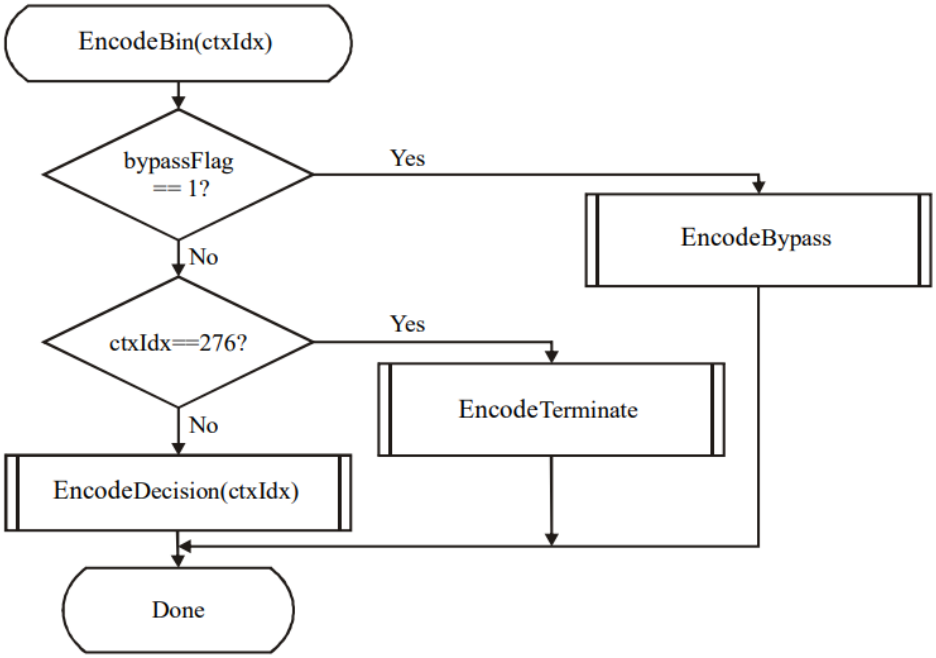
Her context model, belirli bir context ile ilişkilendirilir ve p ve mps olmak üzere iki olasılık değerini korur.

- Olasılık p, 1 değerine sahip bir sembolle karşılaşma olasılığını temsil ederken, mps (en olası sembol), oluşma olasılığı en yüksek olan sembolü (0 veya 1) temsil eder.
- Kodlama işlemi sırasında encoder, işlenmekte olan geçerli SE için uygun context modele başvurur. 1 değerine sahip bir sembolle karşılaşma olasılığını tahmin etmek için context modelden p olasılığını kullanır. Gerçek sembol mps ile eşleşirse, p olasılığı ile kodlanır. Aksi takdirde, (1 - p) olasılığı ile kodlanır.
- Decoder tarafında, 1 değerine sahip bir sembolle karşılaşma olasılığını tahmin etmek için aynı context modeller kullanılır. Kod çözücü, gerçek sembolü belirlemek için tahmini olasılığı alınan kodlanmış değerle karşılaştırarak ters işlemi gerçekleştirir.
- Context model bloğu, olasılık tahminlerini kodlama ve kod çözme sırasında gözlenen sembollere dayalı olarak uyarlar. Bu uyarlama süreci, modelin girdi verilerinin belirli istatistiksel özelliklerine uyum sağlamasına yardımcı olarak sıkıştırma verimliliğinin artmasına yol açar.

3.1.3 Binary aritmetik kodlayıcı

CABAC'ta (Bağlama Uyarlamalı İkili Aritmetik Kodlama), Aritmetik kodlama bloğu, girişlerin ikili sembollerini (bitlerini) verimli kayıpsız sıkıştırmaya izin verecek şekilde kodlamaktan sorumludur. Aritmetik kodlama, veri sıkıştırmada bir mesajı (bir sembol dizisi), sembollerin olasılık dağılımı göz önüne alındığında, mesajı temsil etmek için gereken bit sayısını en aza indirecek şekilde kodlamak için kullanılan bir entropi kodlama biçimidir. Aritmetik kodlamanın arkasındaki temel fikir, mesajı tek bir kesir olarak temsil etmektir; burada kesir, belirli bir değer aralığında yer alacak şekilde seçilir. Mesajdaki her sembol daha sonra değer aralığı değiştirilerek kodlanır, böylece nihai değer aralığı mesaja benzersiz bir şekilde karşılık gelir. Aritmetik kodlama bloğu, girişleri her seferinde bir sembol olarak kodlamak için bu değer aralığını kullanır. Bir sembol her kodlandığında, değer aralığı, sembolün olasılığına bağlı olarak değiştirilir, öyle ki, sonuncu değer aralığı benzersiz bir şekilde mesajın tamamına karşılık gelir. Aritmetik kodlama ünitesinde gerçekleşen işlemler aşağıda gösterildiği gibidir. Eğer

bir bit için bypassFlag =1 ise bit EncodeBypass fonksiyonu kullanılarak kodlanır. Eğer bitin context = 276 ise EncodeTerminate fonksiyonu kullanılarak kodlanır. Diğer türlü bit EncodeDecision fonksiyonu ile kodlanır.

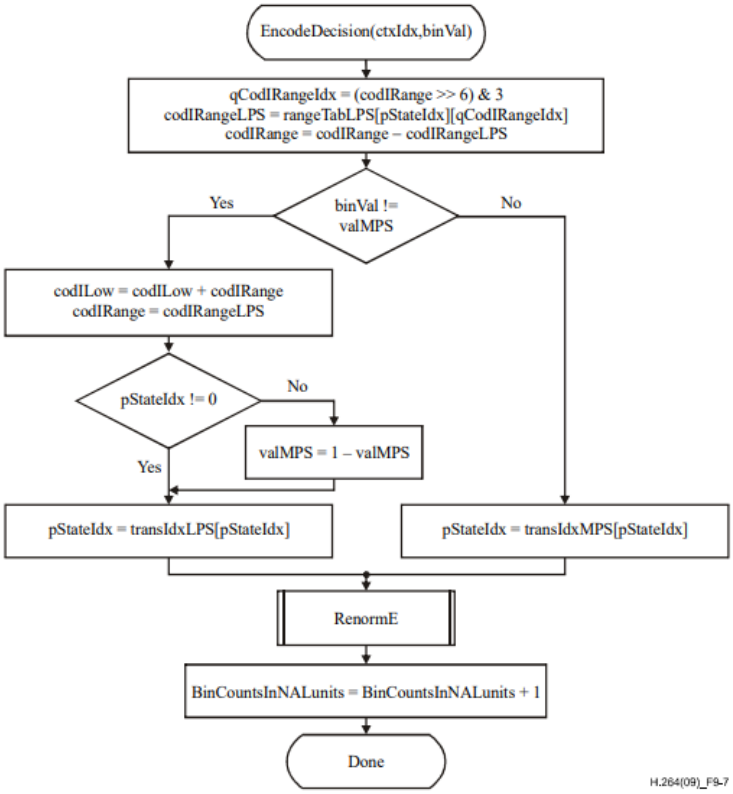


Şekil 3.3 Aritmetik Kodlayıcı İşlemleri[7]

Encode Decision için akış şeması aşağıdaki fotoğrafta belirtilmiştir. Bu sürecin girdileri şunlardır: Context Model ctxIdx, kodlanacak binVal değeri ve codIRange, codILow ve BinCountsInNAUnits değişkenleridir. Bu işlemin çıktıları codIRange, codILow ve BinCountsInNALunits değişkenleridir.

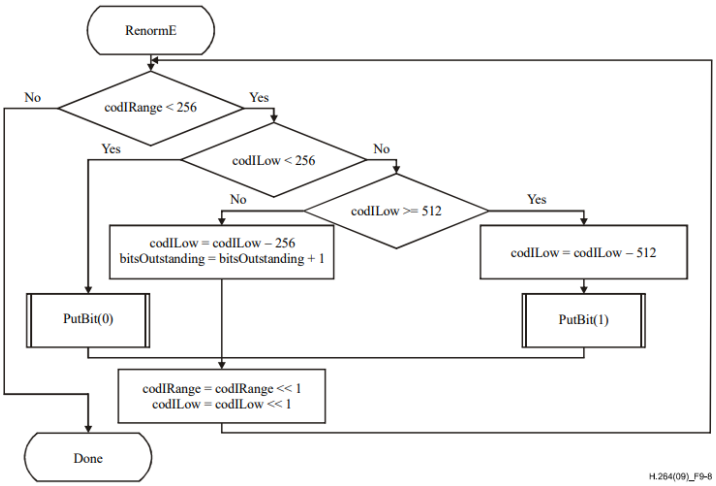
İlk olarak codIRangeLPS parametresi akış şemasında gösterildiği gibi türetilir. codIRange'ın geçerli değeri verildiğinde codIRange, codIRange'ın nicelenmiş bir değerinin qCodIRangeIdx dizinine eşlenir. qCodIRangeIdx değeri ve ctxIdx ile ilişkili pStateIdx değeri, codIRangeLPS'ye atanan rangeTabLPS değişkeninin değerini belirlemek için kullanılır. codirange − codirangeLPS'nin değeri codirange'a atanır.

İkinci bir adımda, binVal değeri, ctxIdx ile valMPS ile karşılaştırılır. binVal, valMPS'den farklı olduğunda, codIRange, codILow'a eklenir ve codIRange, codIRangeLPS değerine eşit olarak ayarlanır. Kodlanmış karar verildiğinde, durum geçişi gerçekleştirilir. codIRange'ın geçerli değerine bağlı olarak yeniden normalleştirme gerçekleştirilir. Son olarak, BinCountsInNAUnits değişkeni 1 artırılır.



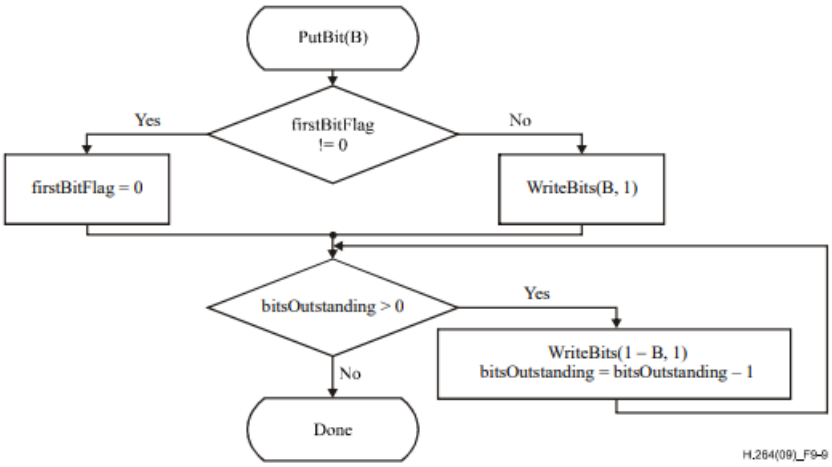
Şekil 3.4 Encode Decision için Akış Şeması[7]

CABAC'ta yeniden normalleştirme işlemi için giriş değişkenleri arasında codIRange, codILow, firstBitFlag ve bitsOutstanding bulunur. Bu işlemin çıktısı, RBSP'ye (Raw Byte Sequence Payload) yazılan sıfır veya daha fazla biti ve codIRange, codILow, firstBitFlag ve bitsOutstanding'in güncellenmiş değerlerini içerir.



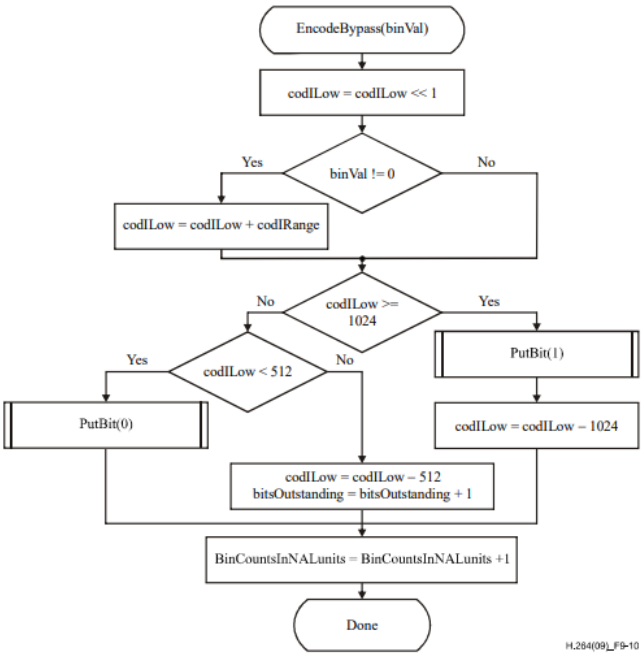
Şekil 3.5 Renormalization Fonksiyonu için Akış Şeması[7]

PutBit() fonksiyonu tek bir bitin bir bit akışına yazılmasına izin verir. Yazılacak bir sonraki bitin konumunu takip eder. Bunu, bit akışına belirli bir değere sahip belirli sayıda bit yazan WriteBits() işlevini kullanarak yapar.



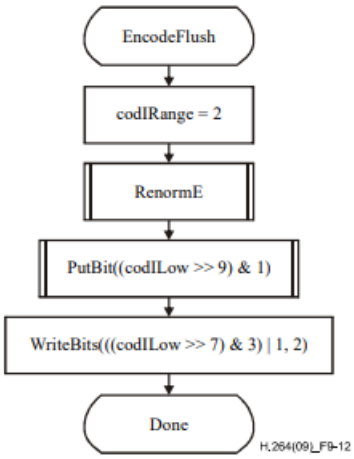
Şekil 3.6 PutBit(B) Fonksiyonu için Akış Şeması [7]

Bu işlemin girdileri binVal, codILow, codIRange, bitsOutstanding ve BinCountsInNALunits parametreleridir. Akış şemasının çıktısı RBSP'ye ve güncellenmiş codILow, bitsOutstanding ve BinCountsInNAUnits değişkenlerine yazılan bir bittir. Bu kodlama işlemi, baypasFlag'ı 1'e eşit olan tüm ikili kararlar için geçerlidir.



Şekil 3.7 EncodeBypass(binVal) Fonksiyonu için Akış Şeması[7]

Kodlanacak binVal değeri 1'e eşit olduğu zaman CABAC kodlaması sonlandırılır ve Aşağıda belirtilmiş olan EncodeFlush fonksiyonu uygulanır. EncodeFlush fonksiyonunda WriteBits(B,N) tarafından yazılan son bit 1'e eşittir. end_of_slice_flag kodlanırken, bu son bit rbsp_stop_one_bit olarak yorumlanır.



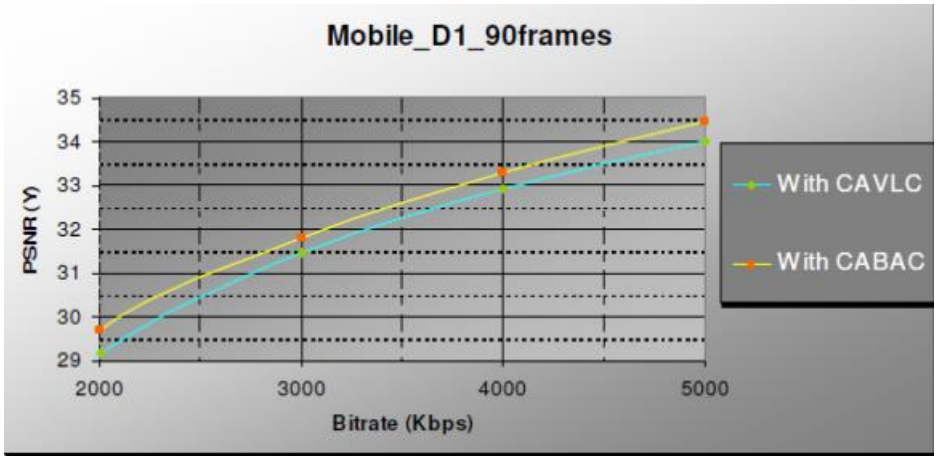
Şekil 3.8 EncodeFlush Fonksiyonu için Akış Şeması [7]

3.2 CABAC ve CAVLC Karşılaştırması

H.264 kuantize dönüşüm katsayılarını kodlamak için CAVLC ve CABAC olmak üzere iki farklı entropi kodlama yöntemi içerir. CABAC sıkıştırma algoritması, CAVLC sıkıştırma algoritmasından belirli profillerde, ana ve yüksek profillerde, daha iyi sıkıştırma performansı sunar[1]. Sıkıştırma performansının daha iyi olmasının sebeplerinden bazıları aşağıda belirtilmiştir:

- CAVLC, katsayı seviyelerine bağlı olarak sıfır ve +/-1 katsayılarını işler ve toplamda oluşan sıfır ve +/-1 sayıları kodlanmıştır. Diğer katsayılar için ise seviyeleri kodlanmıştır. Artık katsayıların bağlama uyarlanabilir VLC'si, çalışma uzunluğu kodlamasını kullanır. CABAC ise aritmetik kodlamayı kullanır. Ayrıca, iyi bir sıkıştırma elde etmek için, her sembol elemanı için olasılık modeli güncellenir. Hem MV hem de artık dönüşüm katsayıları CABAC tarafından kodlanır. [5]

- Uyarlanabilir olasılık modelleri kullanılır ve basit ve hızlı uyarlama mekanizması için ikili aritmetik kodlama ile sınırlıdır. Bağlamlar kullanılarak sembol bağıntılarından yararlanılır.
- CABAC, hesaplama açısından daha karışık olmasına rağmen sıkıştırma performansında CAVLC'ye göre %10 oranında daha verimli olur.[5] CABAC, aşağıdaki şekilde de görüldüğü üzere aynı PSNR[6] için CAVLC'ye karşılaştırıldığı zaman bit oranında %10 ile %15 arasında azalma sağlıyor .



Şekil 3.9 CABAC ve CAVLC karşılaştırması[6]

4. TASARIM

Bu kısım CABAC entropi kodlama algoritmasının MATLAB ve RTL tasarımlarıyla ilgili detayları içermektedir. MATLAB dosyaları, MATLAB modelinin doğrulanması, MATLAB modelinin coverage sonuçları, RTL şematikler ve RTL simülasyon sonuçları gösterilecektir.

4.1 MATLAB Modeli

4.1.1 MATLAB model dosyaları

Aşağıdaki görsel CABAC MATLAB modelinin dosyalarını göstermektedir. Toplam 41 dosyadan oluşan MATLAB modelinin dosyalarının açıklamaları aşağıde verilmiştir.

Şekil 4.1 CABAC MATLAB modelinin dosyaları

- **B_slice_bin.m:** B Slice için mb_type syntax elementi için binarization yapan fonksiyonu içerir.
- **B_sub_mb_bin.m:** B Slice için sub_mb_type syntax elementi için binarization yapan fonksiyonu içerir.
- **Binarizationv2.m:** Binarization bloğu için üst seviye tasarımı içeren fonksiyondur. Girdideki syntax elemente göre gerekli binarization yöntemi/fonksiyonu çalıştırılır.
- **BlockCat_ctx.m:** ctxIdx değeri ctxBlockCat ile belirlenen syntax elementlerden significant_coeff_flag ve last_significant_coeff_flag için ctxIdx belirleme fonksiyonudur.
- **CABACv2.m:** CABAC Top Level tasarım dosyasıdır. Gerekli blok bağlantılarını içerir.
- **Coded_block_flag_ctx.m:** Coded_block_flag_ctx.m mbAddrA, mbAddrB ve isIntra girişlerine sahiptir. Mb_type göre belirlenen condTermFlagA,

condTermFlagB ve ctxBlockCat değerlerine dayalı olarak ctxIdxInc dizin artışını hesaplar.

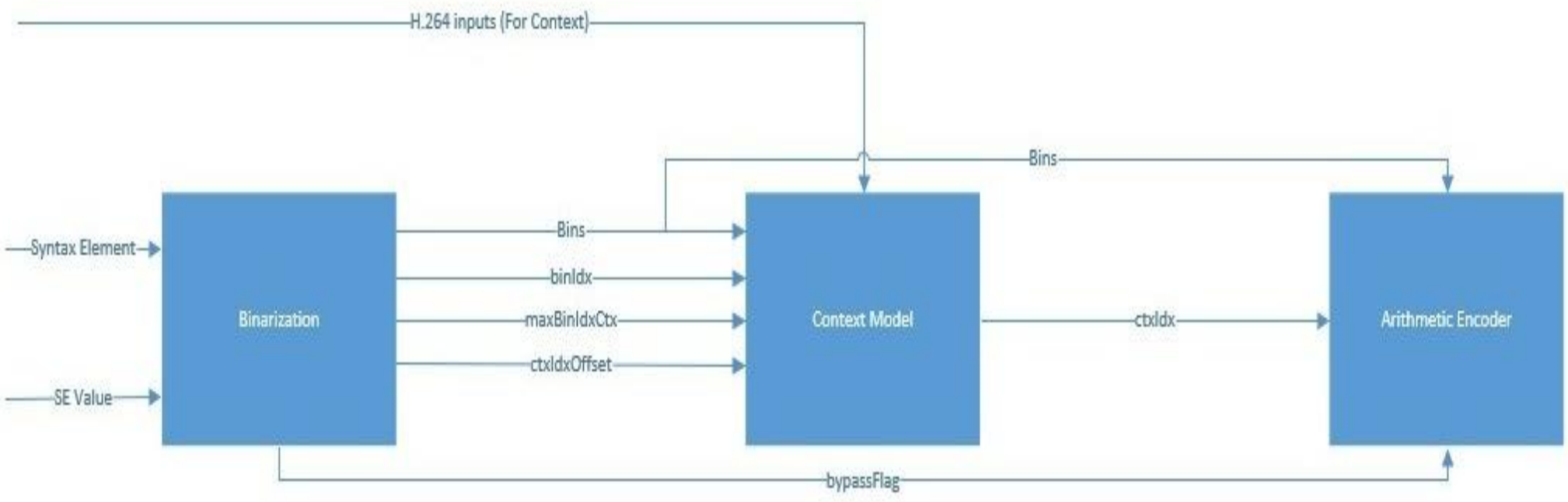
- **Coded_block_pattern_binv2.m:**coded_block_pattern_binv2.m, CodedBlockPatternChroma ve ChromaArrayType girişlerine sahiptir. Bu matlab dosyası ChromaArrayType değişkenin alacağı değere göre belirli durumlarda TU fonksiyonuyla birlikte de suffix değerini hesaplar.
- **Coded_block_pattern_ctx.m** : coded_block_pattern_ctx.m dosyası, mbAddrA, mbAddrB, ctxIdxOffset, cbp_idx ve cbp girişlerine göre ctxIdxInc artış değerini hesaplar. mbAddrA ve mbAddrB'nin kullanılabilirliğini (available) ve türünü(mb_type) kontrol etmeyi, cbp_idx ve mbAddrA.cbp/mbAddrB.cbp üzerinde bitsel işlemler gerçekleştirmeyi ve buna göre condTermFlagA ve condTermFlagB'yi güncellemeyi içerir. ctxIdxInc, bu parametreler kullanılarak hesaplanır ve çıktı olarak döndürülür.
- **Coeff_abs_level_min1_ctx.m:** coeff_abs_level_min1_ctx.m dosyası binIdx, numDecodAbsLevelGt1, numDecodAbsLevelEq1 ve ctxBlockCat girişlerine göre ctxIdx değerini hesaplar. ctxIdxInc değerini belirlemek için koşullu kontroller ve hesaplamalar gerçekleştirir. Bu değer, son ctxIdx'i elde etmek için ctxBlockCat tarafından belirlenen bir offset değeriyle birleştirilir.
- **Context_Modellerv2.m:** Context_Modellerv2 dosyası, sağlanan girdilere ve bunların kombinasyonlarına dayalı olarak context index değerlerini (ctxIdx) belirler. BinIdx dizisinin her öğesi için uygun ctxIdx değerlerini hesaplamak amacıyla farklı durumları kullanır ve çeşitli yardımcı fonksiyonları çağırır.
- **Ctx_init.m:** ctx_init dosyası qp girişine dayalı olarak bağlam öncesi durumları (preCtxState) hesaplayarak ve yapı üyelerine karşılık gelen değerleri atayarak Ictx ve PBctx bağlam yapılarını başlatır.
- **Ctx_using_prior_bin.m:** ctx_using_prior_bin.m dosyası, prior_bin, ctxIdxOffset ve binIdx girişlerine göre cxtIdxInc değerini hesaplar.
- **Encode_Bypassv2:** Encode_Bypassv2.m dosyası girişlere göre codILow değerini güncelleyerek ikili (binary) kodlama gerçekleştirir.

- **Encode_Decisionv2.m:** Encode_Decisionv2.m dosyası girdilere dayalı olarak ikili kodlama (binary) kararları gerçekleştirir. codIRange ve codILow değişkenlerini kodlama kurallarına göre günceller.
- **Encode_init.m:** Encode_init.m dosyası kodlama işlemi için gereken değişkenlerin başlangıç değerlerini ayarlar.
- **Encode_Terminatev2.m**: Encode_Terminatev2.m dosyası, kalan bitleri temizleyerek veya yeniden normalleştirme (renorm) gerçekleştirerek kodlama sürecini sonlandırır ve encode sonucunda ortaya çıkan bit akışını (bitstream) ve güncellenen değişkenleri döndürür.
- **EncodeFlush.m:** EncodeFlush.m dosyası aritmetik kodlama aralığını yeniden normalleştirir (renorm), kalan bitleri kodlar, bir bitiş işaretçisi ekler ve bayt hizalı bir çıktı sağlamak için bit akışını doldurur.
- **Exp_Golomb.m:** Exp_Golomb.m dosyası, Exp-Golomb kodlama şemasını kullanarak bir sembolü kodlar ve kodlanan bitleri bir dizi olarak döndürür.
- **FL.m:** FL dosyası, belirli bir kod_sayısı için belirli bir cMax aralığında, kod_num'u ikili tabana (binary) dönüştürüp ters çevirerek sabit uzunlukta bir ikili (binary) kod üretir.
- **I_slice_bin.m:** I_slice_bin dosyası SynVal'in belirli değerlerini I-slice kodlaması için ilgili ikili kodlarına (binary code) eşler.
- **init_tableI.m:** Inıt_tableI.m dosyası bir i girişi alan ve m ve n olmak üzere iki değişken döndüren init_tableI fonksiyonunu tanımlar. İşlev içinde, önceden tanımlanmış bir dizi değer içeren bir matris olan bir başlatma tablosu init_table_I vardır (I-slice için).
- **init_tablePB.m:** Inıt_tableP.m dosyası bir i girişi alan ve m ve n olmak üzere iki değişken döndüren init_tableP fonksiyonunu tanımlar. İşlev içinde, önceden tanımlanmış bir dizi değer içeren bir matris olan bir başlatma tablosu init_table_P vardır (P-slice ve B-slice için).
- **intra_chroma_pred_mode_ctx.m:** intra_chroma_pred_mode_ctx.m dosyası (mbAddrA ve mbAddrB) özelliklerine ve kullanılabilirliğine (available) bağlı olarak iki koşul bayrağının (condTermFlag) değerlerini belirler. Bu işaretler daha sonra bağlam indeksi artış (ctxIdxInc) değerini hesaplamak için kullanılır.

- **mb_field_decoding_flag_ctx.m:** mb_field_decoding_flag_ctx.m dosyası (mbAddrA ve mbAddrB) özelliklerine ve kullanılabilirliğine (available) bağlı olarak iki koşul bayrağının (condTermFlag) değerlerini belirler. Bu işaretler daha sonra bağlam indeksi artış (ctxIdxInc) değerini hesaplamak için kullanılır.
- **mb_qp_delta_bin.m:** mb_qp_delta_bin.m dosyası verilen bir değerin işaretini (SynVal) belirler ve bu işarete SynVal'in mutlak değerine göre bir sembol hesaplar. Unary fonksiyonu daha sonra bu sembol ile çağrılır ve elde edilen değer kod değişkenine atanır.
- **mb_qp_delta_ctx.m**: mb_qp_delta_ctx.m dosyası mbAddrA'daki mb_qp_delta değerine bağlı olarak bağlam indeksi artış (ctxIdxInc) değerini belirler.
- **mb_skip_flag_ctx.m:** mb_skip_flag_ctx.m dosyası bağlam indeksi artış (ctxIdxInc) değerini mbAddrA ve mbAddrB'nin mb_skip_flag özelliğine göre belirler.
- **mb_type_ctx.m**: mb_type_ctx.m dosyası, mbAddrA ve mbAddrB'nin özelliklerine ve ctxIdxOffset değerine bağlı olarak iki koşullu bayrağın (condTermFlagA ve condTermFlagB) değerlerini belirler. Bu parametrelerin toplamı da bağlam indeksi artış değerini (ctxIdxInc) verir.
- **mvd_ctx.m:** mvd_ctx.m dosyası mbAddrA ve mbAddrB'nin kullanılabilirliğine (available) bunların hareket vektörü farklılıklarına (mvd) ve mb_aff_frame_flag, compIdx ve mb_field ile ilgili ek koşullara dayalı olarak bağlam indeksi artışını (ctxIdxInc) hesaplar. Ortaya çıkan ctxIdxInc değeri, sonraki işlemlerde bağlam indeksi offsetini (ctxIdxOffset) belirler.
- **P_SP_slice_bin.m:** P_SP_slice_bin.m dosyası SynVal değerine bağlı olarak bir ikili kod (binary code) üretir. İlgili ikili kod, SynVal'ın değeri tarafından belirlenir.
- **P_SP_sub_mb_bin.m:** P_SP_sub_mb_bin.m dosyası SynVal değerine dayalı olarak belirli bir ikili kod (binary code) atar.
- **PutBit.m:** putbit.m dosyası B'nin değerine dayalı olarak bit akışı (bitstream) değişkenine bitler ekleyerek bir ikili bit akışı oluşturur. Ayrıca, döngünün her yinelemesinde firstBitFlag'ı günceller ve bitsOutstanding'i azaltır. Elde

edilen bit akışı, güncellenmiş firstBitFlag ve bitsOutstanding ile birlikte işlevin çıktısı olarak döndürülür.

- **ref_ctx.m:** ref_ctx.m dosyası mbAddrA ve mbAddrB'nin kullanılabilirliği , türü, alanı, sub_mb_type ve ref_idx'ini içeren çeşitli koşullara dayalı olarak bir bağlam dizini artışını (ctxIdxInc) hesaplar. Daha sonra ise toplanan condTermFlagA ve condTermFlagB parametreleri ctxIdxInc değerini belirler.
- **RenormE.m:** RenormE.m dosyası aritmetik kodlama için bir yeniden normalleştirme işlemi gerçekleştirir. codIRange ve codILow değerlerine dayalı olarak PutBit işlevini çağırarak ikili bir bit akışı oluşturur. İşlev ayrıca döngü içinde codIRange, codILow, firstBitFlag ve bitsOutstanding'i günceller.
- **SI_slice_bin.m:** SI_slice_bin.m dosyası SynVal değerine dayalı olarak bir ikili kod üretir.
- **transform_size_8x8_flag_ctx:** transform_size_8x8_flag_ctx.m dosyası komşu makrobloklar mbAddrA ve mbAddrB'nin değerlerine dayalı olarak, dönüşüm boyutu 8x8 bayrağı için bağlam artışını (ctxIdxInc) belirler. CtxIdxInc, condTermFlagA ve condTermFlagB toplamına eşittir.
- **TU.m:** TU.m dosyası cMax ve code_num değerlerine dayalı olarak değişken uzunluklu bir kod üretir.
- **UEGk.m:** UEGk.m dosyası uCoff,signValFlag ve k parametrelerine dayalı değişken uzunluklu bir kod kullanarak bir SynVal giriş değerini kodlar.
- **Unarym:** Unary.m dosyası code_num girişine bağlı olarak unary code üretir.
- **Dosyaokutma.m:** Dosyaokutma.m dosyası, ilgili video kaynağının sayısal verilerini CSV dosyası yardımıyla okur ve ardından CABAC sıkıştırma algoritması için her bir veri satırını işler. İşlenen veriler sürekli olarak bitstream üzerine eklenir. İki noktada ,başlatmadan sonra ve tüm veriler işlendikten sonra, bitstream onaltılık tabana (hexadecimal) dönüştürür ve ilgili videonun kodlanmış halini bir .txt uzantılı dosyaya yazar.



Şekil 4.2 CABAC MATLAB modeli blok diyagramı

4.1.2 MATLAB modelinin doğrulanması

- MATLAB Modelini çalıştırmak için öncelikle referans yazılım olan JM yazılımından kodlanacak verileri toplamak gerekmektedir. Bunun için yazılımda birtakım düzenlemeler yapılarak gerekli veriler bir dosyaya yazdırılır. Bu düzenlemeler kodlanan her SE için yazılmış fonksiyonların içerisine bir dosyaya yazdırma fonksiyonu ile birlikte kodlanacak SE, kodlanacak SE'nin değeri, ctxIdx hesaplanabilmesi için gerekli bağlam girdileri (mbAddrA, mbAddrB vs.), Slice türü gibi gerekli parametrelerin yazdırılmasıdır

Örnek bir kod düzenlemesi aşağıda gözükmektedir

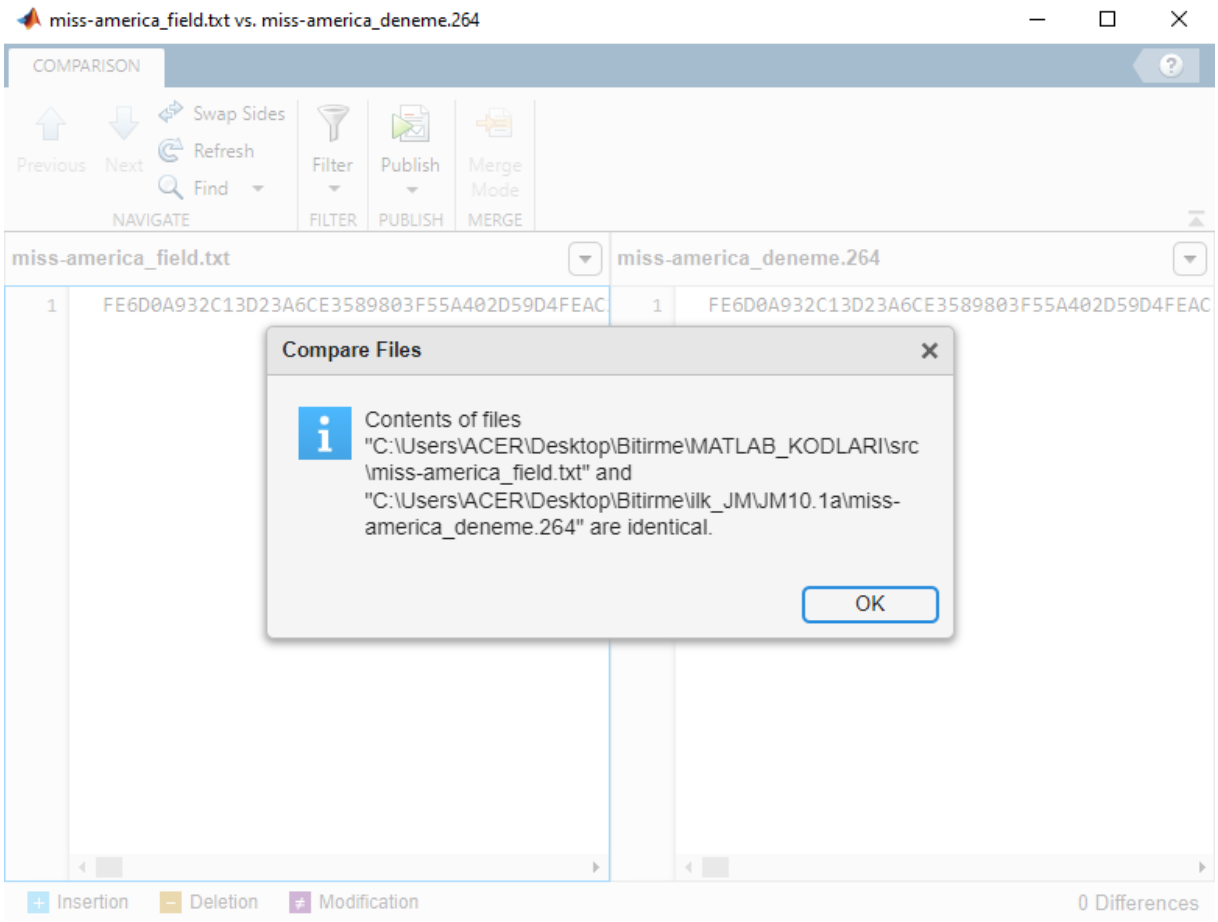
```
void writeMB_transform_size_CABAC(SyntaxElement *se, EncodingEnvironmentPtr eep_dp)
{
  int a, b;
  int act_ctx = 0;
  int act_sym;

  MotionInfoContexts *ctx         = (img->currentSlice)->mot_ctx;
  Macroblock         *currMB      = &img->mb_data[img->current_mb_nr];


  b = (currMB->mb_available_up == NULL) ? 0 : currMB->mb_available_up->luma_transform_size_8x8_flag;
  a = (currMB->mb_available_left == NULL) ? 0 :currMB->mb_available_left->luma_transform_size_8x8_flag;

  act_ctx     = a + b;
  act_sym     = currMB->luma_transform_size_8x8_flag;
  se->context = act_ctx; // store context

  FILE* fptr;
  if (img->cabac_yaz) {
      fptr = fopen("girdiler.txt", "a");

      fprintf(fptr, "25;%d;%d;0;0;0;0;0;0;0;%d;0;0;0;%d;0;0;0;0;0;0;0;%d;0;0;0;0;0;0;0;0;0;0;0;0;0;%d;0;0\n", currMB->luma_transform_size_8x8_flag, currMB->mb_available_left

      fclose(fptr);
  }
  biari_encode_symbol(eep_dp, (signed short) (act_sym != 0), ctx->transform_size_contexts + act_ctx );
}
```

Şekil 4.3 Gerekli girdi parametrelerini dosyaya yazdıran kod düzenlemesi

- Bu veriler dosyaya yazıldıktan sonra MATLAB Modeli çalışmaya hazırdır. Dosyaokutma.m adlı MATLAB kodunda gerekli verileri içeren dosya okutulur. CSV dosyası formatında okutulan dosyanın her sütunu kendisine karşılık gelen değişkene atanır. Kodun çalışması bittiğinde kodlanmış, yani sıkıştırılmış, veriler bir dosyaya hexadecimal (on altılık) tabanda yazdırılır.
- JM yazılımı sonucu oluşan çıktıların ve MATLAB modelimizin çıktısını daha rahat karşılaştırmak adına, JM yazılımında dosyaya yazma işlemlerinde de birtakım değişiklikler yapmak gerekmektedir. Bunlardan

ilki video ile ilgili parametreler gibi verileri içeren header bitlerinin dosyaya yazdıran fonksiyonları yorum satırına almaktır. Çünkü bu headerların yazdırılması/kodlanması CABAC modeline dahil olan bir işlem değildir. Bir diğeri ise çıktı dosyasına "binary" formatta yazan fwrite fonksiyonunu, "text" biçiminde yazan fprintf ile değiştirmektir. Bu sayede MATLAB modelimizden çıkan txt dosyası ile JM yazılımından çıkan dosyayı ister karşılaştırma uygulamaları, isterse MATLAB'da yazılacak yeni bir kod ile karşılaştırmak çok daha basit hale gelecektir

- Aşağıda MATLAB Modelinin çıktısı ile JM yazılımından alınan çıktının MATLAB üzerinde karşılaştırılması gözükmektedir. Txt uzantılı dosya MATLAB modelinden, 264 uzantılı dosya ise JM yazılımından alınan çıktılardır. Karşılaştırma MATLAB uygulamasının "Compare" sekmesinden yapılmaktadır. Figürde de görüleceği üzere iki dosyanın çıktıları tamamen uyuşmaktadır.



Şekil 4.4 JM modeli ve MATLAB modeli çıktılarının karşılaştırılması

4.1.3 MATLAB modelinin coverage analizi

Aşağıda paylaşacağımız coverage sonuçları yaklaşık 3500 frame'in test edilmesinden sonra alınmış sonuçlardır. Genel olarak sonuçların yüksek olduğu gözlenmiştir. En temel 7 MATLAB dosyası için coverage sonuçları aşağıda paylaşılmıştır.

Total lines in function	28
Non-code lines (comments, blank lines)	11
Code lines (lines that can run)	17
Code lines that did run	17
Code lines that did not run	0
Coverage (did run/can run)	100.00 %

Şekil 4.5 CABAC.m dosyasının coverage sonucu

- CABAC Üst Level tasarımını içeren CABAC.m dosyasının coverage sonucunu görmekteyiz. Bu dosyada gerekli blok bağlantılarını içerdiğinden dolayı kodların tamamının çalışması beklenen bir sonuçtur.

Total lines in function	569
Non-code lines (comments, blank lines)	56
Code lines (lines that can run)	513
Code lines that did run	279
Code lines that did not run	234
Coverage (did run/can run)	54.39 %

Şekil 4.6 Binarization bloğunun coverage sonucu

- Binarization Bloğunun coverage sonuçları düşük bir oran gibi gözükebilir ancak bu biraz yanıltıcıdır. Aslında detaylı

incelendiğinde tüm SE'lerin binarization fonksiyonlarının çalıştığı gözükmektedir. Çalışmayan satırlar ise aynı SE'in farklı versiyonlarıdır. Örneğin coeff_abs_level_minus1 SE'i için ctxBlockCat 5 için olan kod çalışmış ancak 13 için çalışmamıştır. Bu satırlar toplandığında yüksek bir oranın çalışmadığı kanısına varılabilir ancak çalışan satırlar ile çalışmayan satırlar arasındaki tek fark değişkenlere atanan sayılardır. Dolayısıyla bütün binarization yöntemlerinin çalıştırıldığı söylenilebilir.

Total lines in function	227
Non-code lines (comments, blank lines)	8
Code lines (lines that can run)	219
Code lines that did run	174
Code lines that did not run	45
Coverage (did run/can run)	79.45 %

Şekil 4.7 Context model bloğunun coverage sonucu

- Context Model bloğunda da binarization bloğuna benzer bir durum gözlenmektedir ancak burada coverage sonucumuzun daha yüksek çıkmasının sebebi fonksiyonların tek satır kaplamasındandır. Çalışmayan 45 satırın çoğunluğu yine aynı SE'lerin farklı versiyonlarıdır. Detaylı bakıldığında tek bir SE hariç tamamı için context hesaplamasının yapıldığı görülmektedir.

Total lines in function	109
Non-code lines (comments, blank lines)	72
Code lines (lines that can run)	37
Code lines that did run	37
Code lines that did not run	0
Coverage (did run/can run)	100.00 %

Şekil 4.8 Encode_decision fonksiyonunun coverage sonucu

- Encode_Decision Fonksiyonunda da %100 sonucuna ulaştığımızı gözlemlemekteyiz. Birçok frame test etmenin getirisi olarak özellikle I slice ve P/B slice olarak iki ihtimale ayırdığımız bu fonksiyonun satırlarının tamamının çalıştığını gözlemledik.

Total lines in function	23
Non-code lines (comments, blank lines)	3
Code lines (lines that can run)	20
Code lines that did run	20
Code lines that did not run	0
Coverage (did run/can run)	100.00 %

Şekil 4.9 Encode_Bypass fonksiyonunun coverage sonucu

- Encode_Bypass modülümüz de yine %100 çalışma oranına ulaşmıştır.

Total lines in function	14
Non-code lines (comments, blank lines)	3
Code lines (lines that can run)	11
Code lines that did run	11
Code lines that did not run	0
Coverage (did run/can run)	100.00 %

Şekil 4.10 Encode_terminate fonksiyonunun coverage sonucu

- Diğer Encode modüllerine benzer olarak Encode_Terminate modülümüz de %100 coverage oranına sahiptir.

Total lines in function	18
Non-code lines (comments, blank lines)	1
Code lines (lines that can run)	17
Code lines that did run	17
Code lines that did not run	0
Coverage (did run/can run)	100.00 %

Şekil 4.11 RenormE fonksiyonunun coverage sonucu

- Her Encode modülünde Renormalization adında bir işlem gerçekleştirilmektedir. Bu işlemi yapan RenormE adlı fonksiyonu da bu sebeple koymak istedik. RenormE fonksiyonunda da tüm encode fonksiyonlarında olduğu gibi %100 çalışma oranına ulaşmış durumdayız.

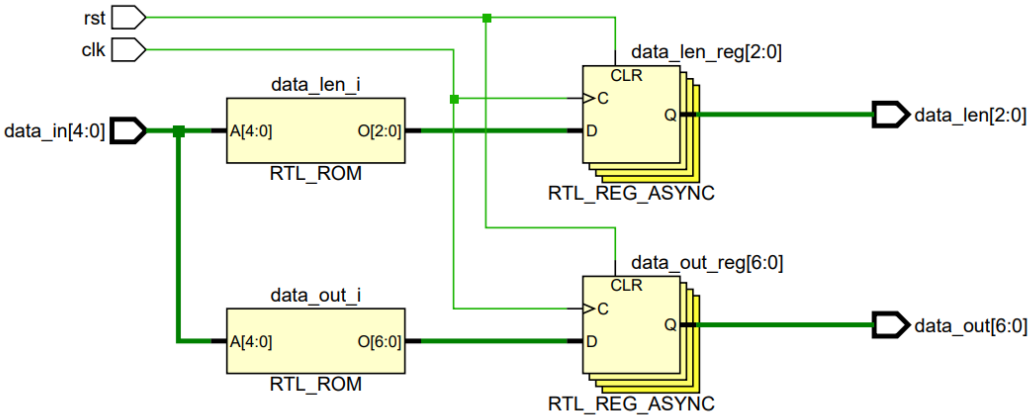
4.2 RTL Tasarım

Aşağıda binarization üst seviye tasarımı ve içerdiği alt bloklara ait RTL tasarımı şematik gösterimleri, simülasyon sonuçları ve kaynak kullanımları gösterilmektedir.

4.2.1 I_Slice_binarization

Aşağıda I_Slice_binarization modülüne ait RTL şematik ve kaynak kullanım sonuçları verilmiştir. Bu sonuçlara göre:

- Kullanılan LUT sayısı:5
- Kullanılan FF sayısı: 10
- Kullanılan I/O portları sayısı: 17

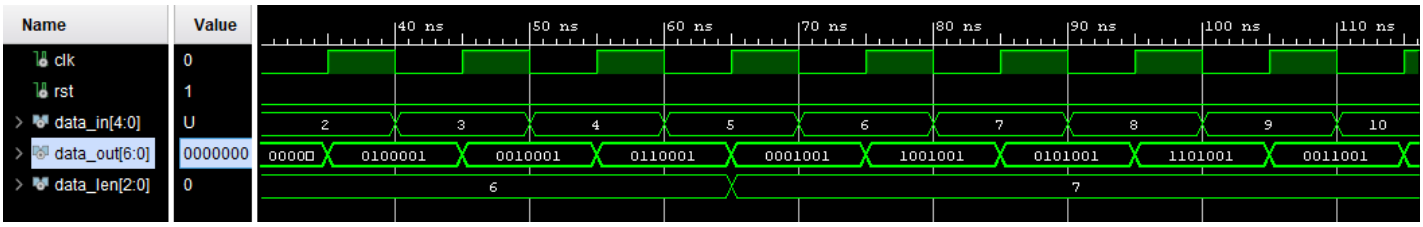


Şekil 4.12 I_Slice_binarization modülüne ait RTL şematik

Resource	Utilization	Available	Utilization %
LUT	5	41000	0.01
FF	10	82000	0.01
IO	17	300	5.67

Şekil 4.13 I_Slice_binarization modülüne ait kaynak kullanım sonuçları

Önceki kısımlarda anlatıldığı gibi simülasyon sonuçlarımız doğru çıktıları verdiği aşağıdaki görselde belirtilmektedir.

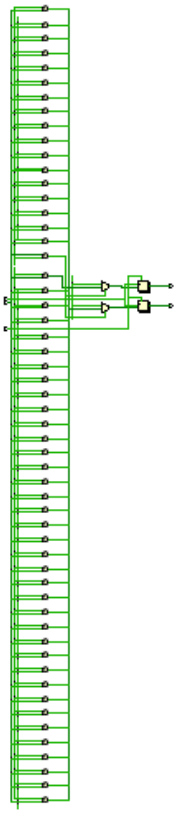


Şekil 4.14 I_Slice_binarization modülüne ait simülasyon sonuçları

4.2.2 Unary binarization

Aşağıda Unary binarization modülüne ait RTL şematik, simülasyon sonucu ve kaynak kullanım sonuçları verilmiştir. Bu sonuçlara göre:

- Kullanılan LUT sayısı:53
- Kullanılan FF sayısı: 60
- Kullanılan I/O portları sayısı: 69

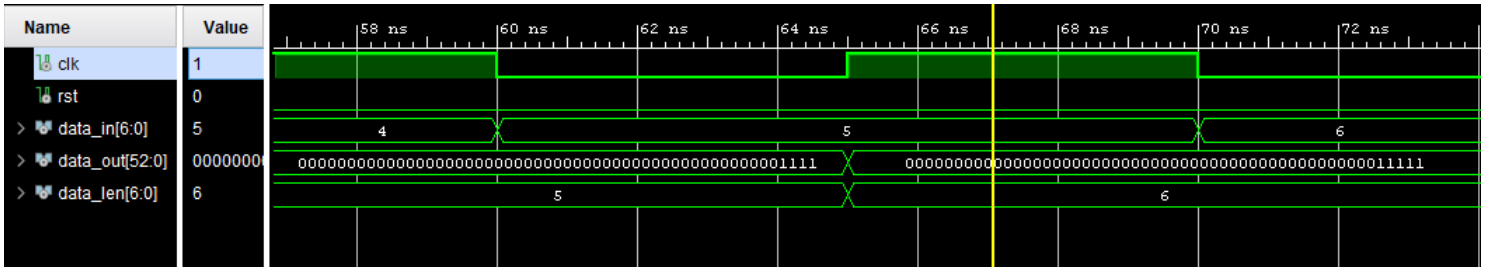


Şekil 4.15 Unary binarization modülüne ait RTL şematik

Resource	Utilization	Available	Utilization %
LUT	53	41000	0.13
FF	60	82000	0.07
IO	69	300	23.00

Şekil 4.16 Unary binarization modülüne ait kaynak kullanım sonuçları

Önceki kısımlarda anlatıldığı gibi simülasyon sonuçlarımız doğru çıktıları verdiği aşağıdaki görselde belirtilmektedir.

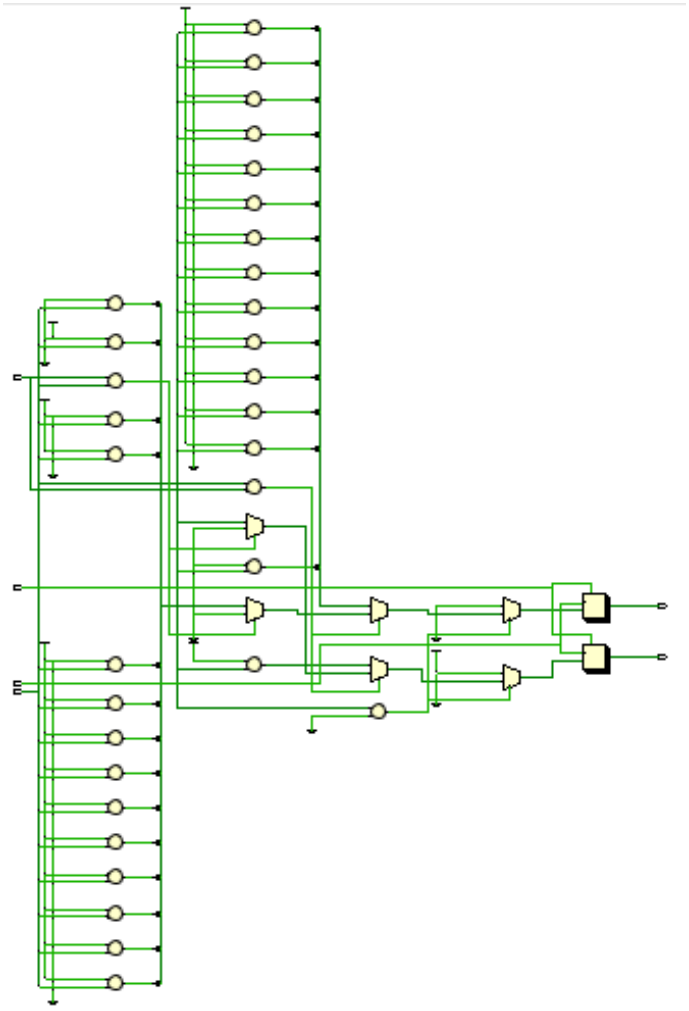


Şekil 4.17 Unary binarization modülüne ait simülasyon sonuçları

4.2.3 TU binarization

Aşağıda TU binarization modülüne ait RTL şematik, simülasyon sonucu ve kaynak kullanım sonuçları verilmiştir. Bu sonuçlara göre:

- Kullanılan LUT sayısı:20
- Kullanılan FF sayısı: 18
- Kullanılan I/O portları sayısı: 28

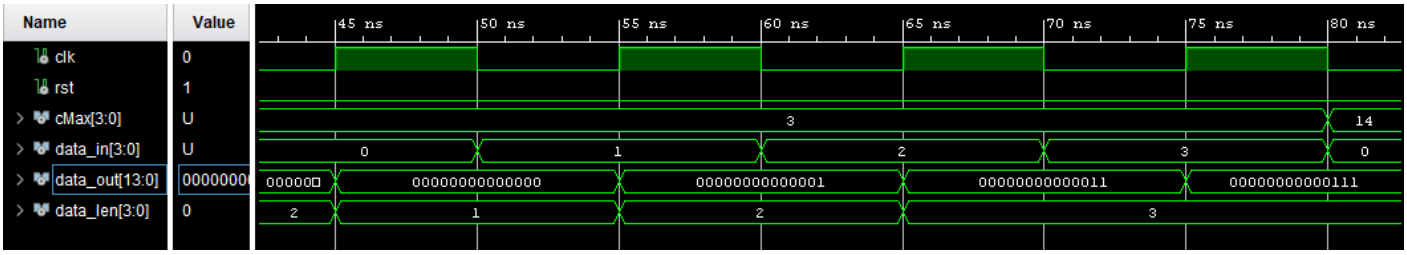


Şekil 4.18 TU binarization modülüne ait RTL şematik

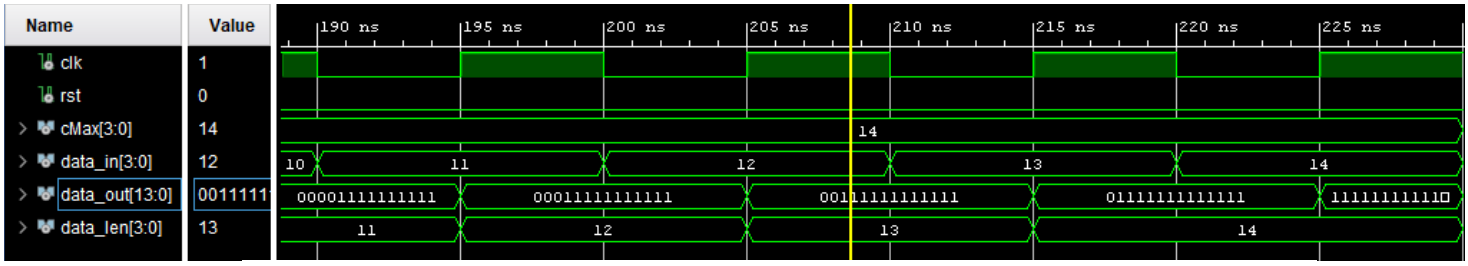
Resource	Utilization	Available	Utilization %
LUT	20	41000	0.05
FF	18	82000	0.02
IO	28	300	9.33

Şekil 4.19 TU binarization modülüne ait kaynak kullanım sonuçları

Önceki kısımlarda anlatıldığı gibi simülasyon sonuçlarımız hem cMax=3 hem de cMax=14 iken doğru çıktıları verdiği aşağıdaki görselde belirtilmektedir.



Şekil 4.20 TU binarization modülüne ait cMax=3 iken simülasyon sonuçları

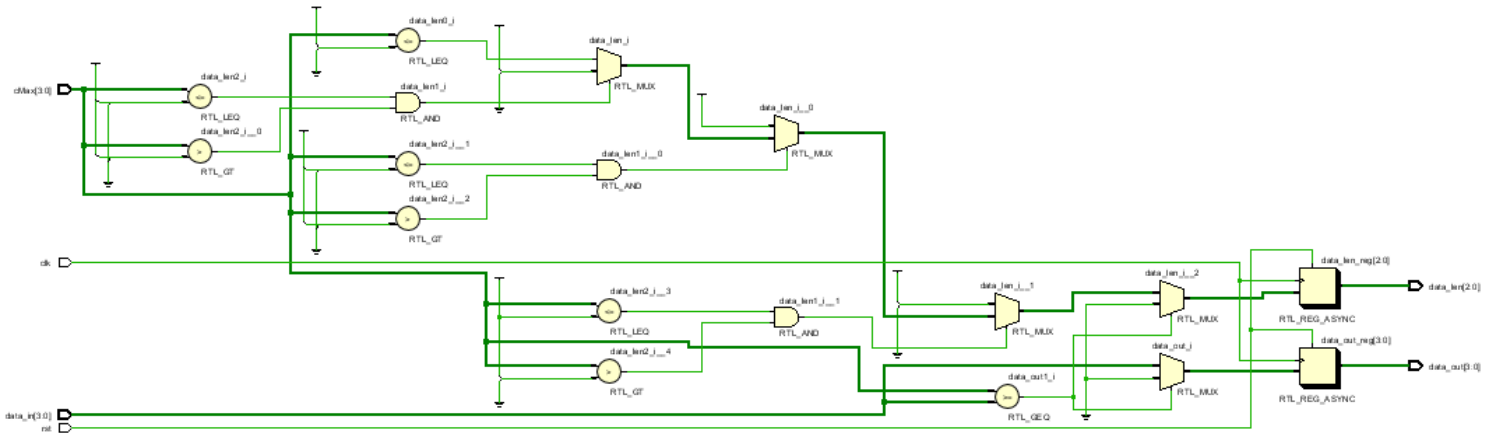


Şekil 4.21 TU binarization modülüne ait cMax=14 iken simülasyon sonuçları

4.2.4 FL binarization

Aşağıda FL binarization modülüne ait RTL şematik, simülasyon sonucu ve kaynak kullanım sonuçları verilmiştir. Bu sonuçlara göre:

- Kullanılan LUT sayısı:5
- Kullanılan FF sayısı: 7
- Kullanılan I/O portları sayısı: 15

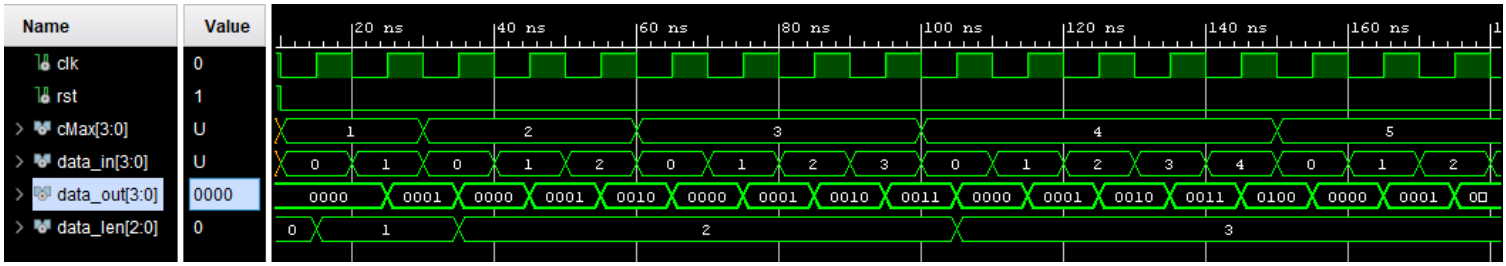


Şekil 4.22 FL binarization modülüne ait RTL şematik

Resource	Utilization	Available	Utilization %
LUT	5	41000	0.01
FF	7	82000	0.01
IO	17	300	5.67

Şekil 4.23 FL binarization modülüne ait kaynak kullanım sonuçları

Önceki kısımlarda anlatıldığı gibi simülasyon sonuçlarımız farklı cMax değerleri verilip çıktıları değerlendirilmiştir, bu değerlendirme sonucunda bu modülün doğru çalıştığı doğrulanmıştır.

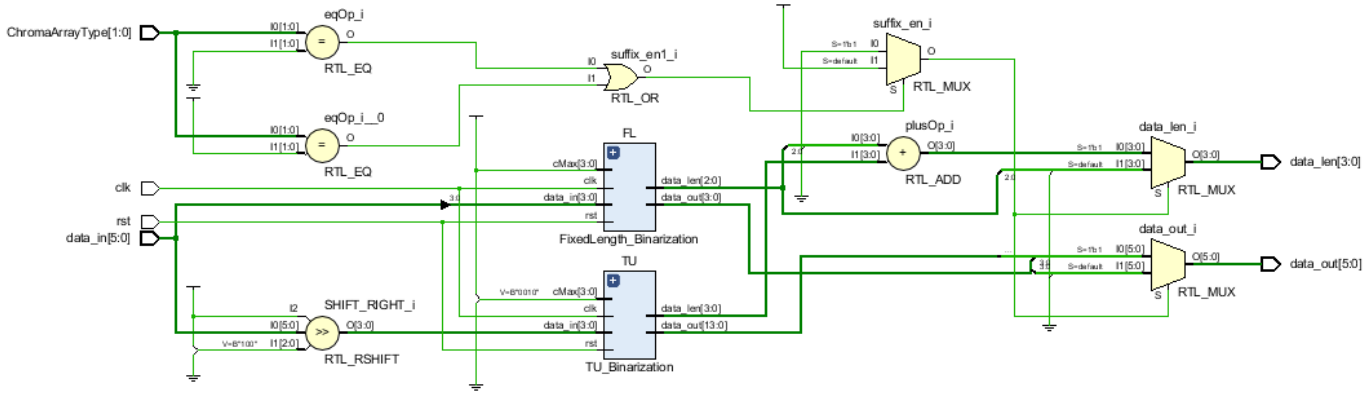


Şekil 4.24 FL binarization modülüne ait simülasyon sonuçları

4.2.5 Coded block pattern binarization

Aşağıda coded block pattern binarization modülüne ait RTL şematik, simülasyon sonucu ve kaynak kullanım sonuçları verilmiştir. Bu sonuçlara göre:

- Kullanılan LUT sayısı:4
- Kullanılan FF sayısı: 8
- Kullanılan I/O portları sayısı: 20

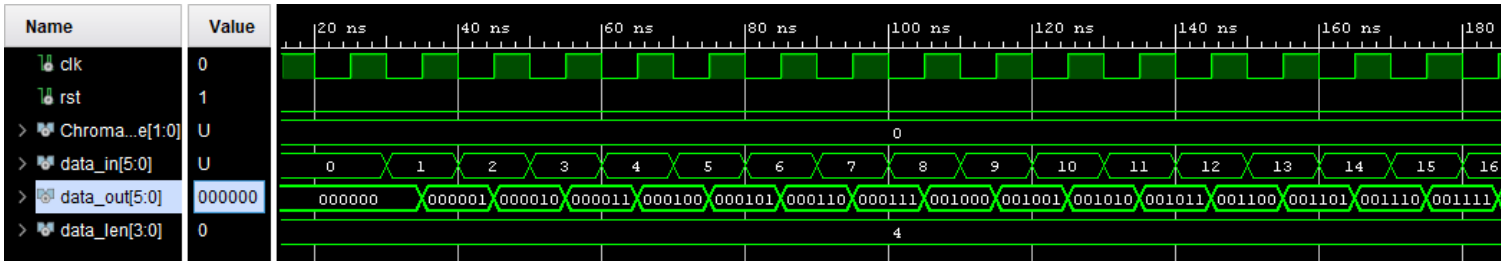


Şekil 4.25 Coded block pattern binarization modülüne ait RTL şematik

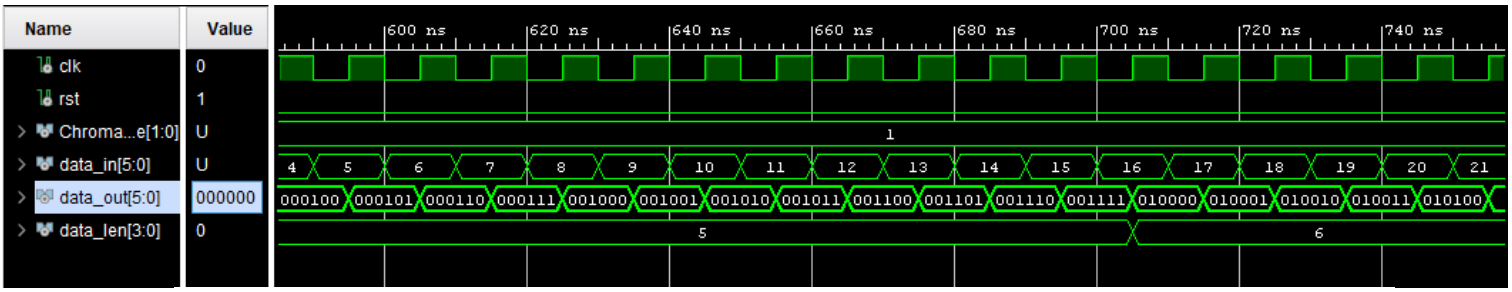
Resource	Utilization	Available	Utilization %
LUT	4	41000	0.01
FF	8	82000	0.01
IO	20	300	6.67

Şekil 4.26 Coded block pattern binarization modülüne ait kaynak kullanım sonuçları

Önceki kısımlarda anlatıldığı gibi simülasyon sonuçlarımız doğru çıktıları verdiği aşağıdaki görselde belirtilmektedir.



Şekil 4.27 Coded block pattern binarization modülüne ait ChromaArrayType=0 iken simülasyon sonuçları

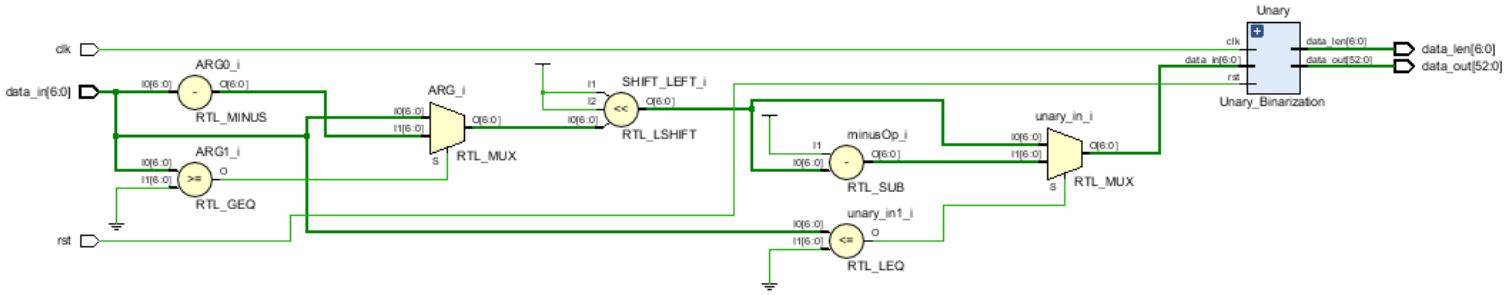


Şekil 4.28 Coded block pattern binarization modülüne ait ChromaArrayType=1 iken simülasyon sonuçları

4.2.6 Mb qp delta binarization

Aşağıda mb qp delta binarization modülüne ait RTL şematik, simülasyon sonucu ve kaynak kullanım sonuçları verilmiştir. Bu sonuçlara göre:

- Kullanılan LUT sayısı:77
- Kullanılan FF sayısı: 60
- Kullanılan I/O portları sayısı: 69

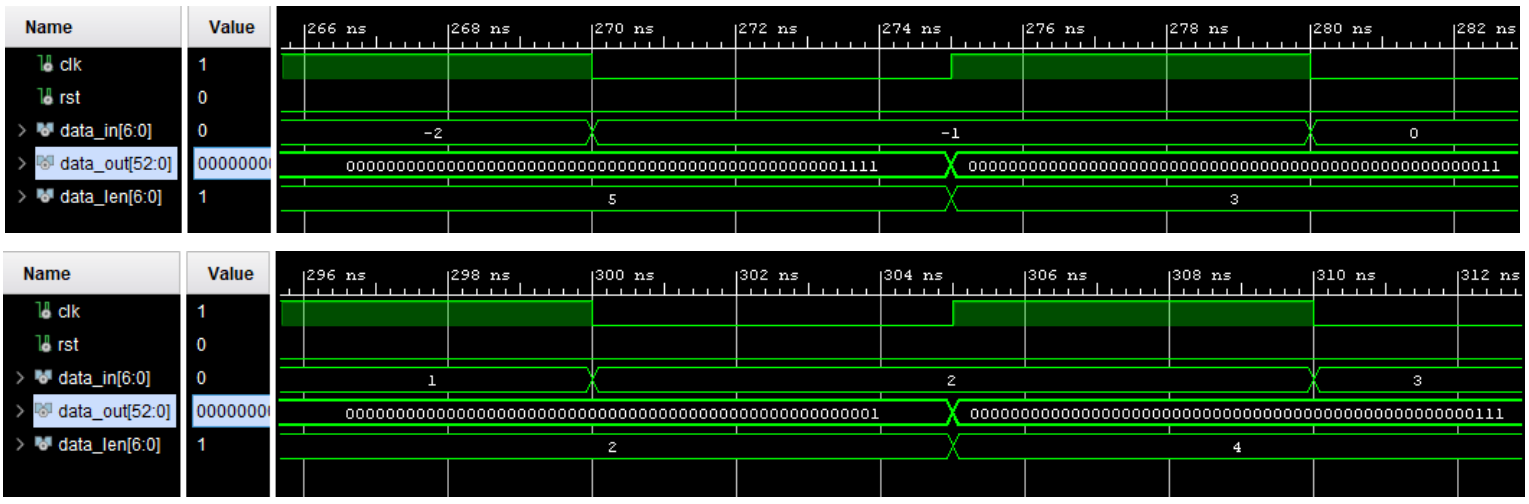


Şekil 4.29 Mb qp delta binarization modülüne ait RTL şematik

Resource	Utilization	Available	Utilization %
LUT	77	41000	0.19
FF	60	82000	0.07
IO	69	300	23.00

Şekil 4.30 Mb qp delta binarization modülüne ait kaynak kullanım sonuçları

Önceki kısımlarda anlatıldığı gibi simülasyon sonuçlarımız doğru çıktıları verdiği aşağıdaki görselde belirtilmektedir.

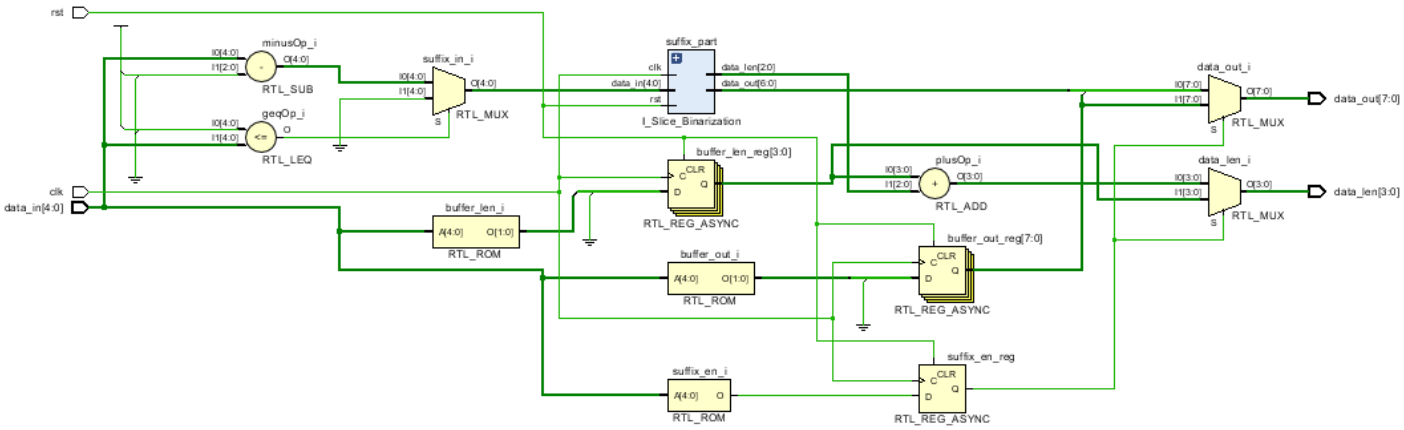


Şekil 4.31 Mb qp delta binarization modülüne ait simülasyon sonuçları

4.2.7 P slice mb type binarization

Aşağıda p slice mb type binarization modülüne ait RTL şematik, simülasyon sonucu ve kaynak kullanım sonuçları verilmiştir. Bu sonuçlara göre:

- Kullanılan LUT sayısı:18
- Kullanılan FF sayısı: 15
- Kullanılan I/O portları sayısı: 19

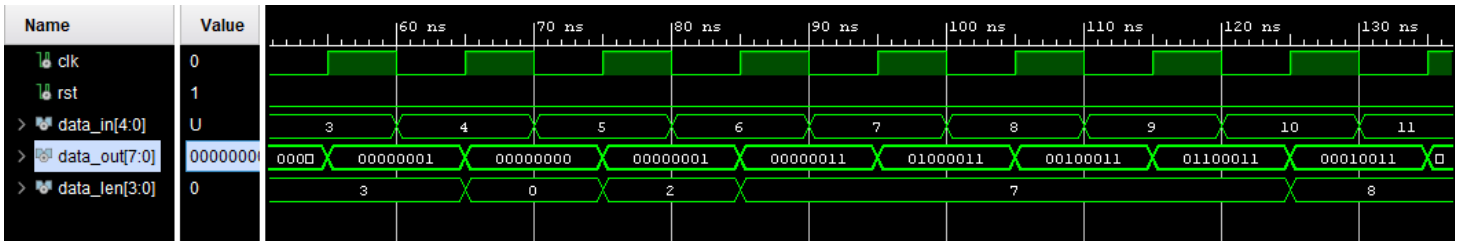


Şekil 4.32 P slice mb type binarization modülüne ait RTL şematik

Resource	Utilization	Available	Utilization %
LUT	18	41000	0.04
FF	15	82000	0.02
IO	19	300	6.33

Şekil 4.33 P slice mb type binarization modülüne ait kaynak kullanım sonuçları

Önceki kısımlarda anlatıldığı gibi simülasyon sonuçlarımız doğru çıktıları verdiği aşağıdaki görselde belirtilmektedir.

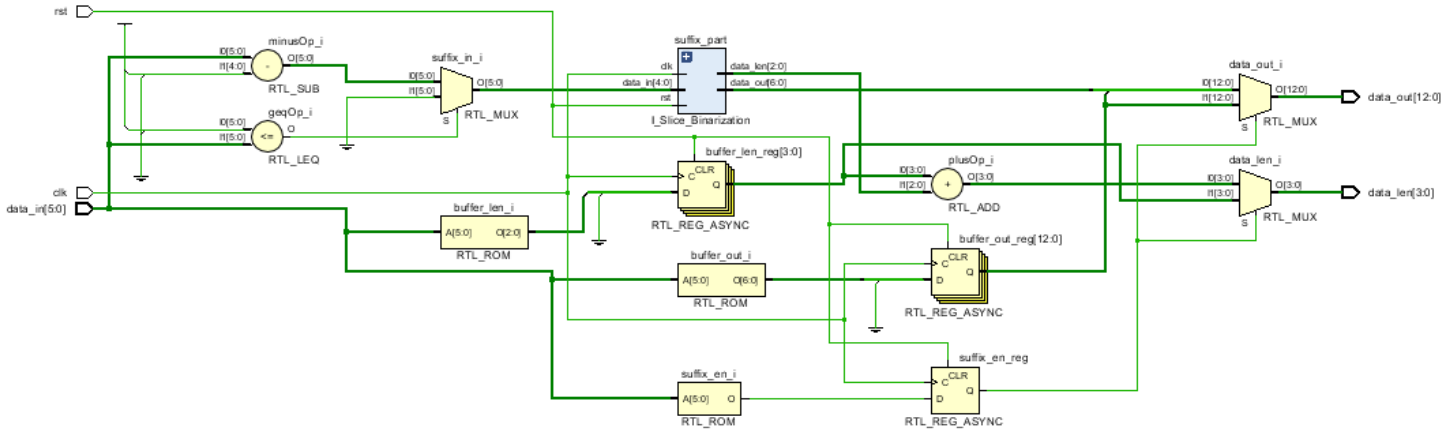


Şekil 4.34 P slice mb type binarization modülüne ait simülasyon sonuçları

4.2.8 B slice mb type binarization

Aşağıda b slice mb type binarization modülüne ait RTL şematik, simülasyon sonucu ve kaynak kullanım sonuçları verilmiştir. Bu sonuçlara göre:

- Kullanılan LUT sayısı:28
- Kullanılan FF sayısı: 21
- Kullanılan I/O portları sayısı: 25

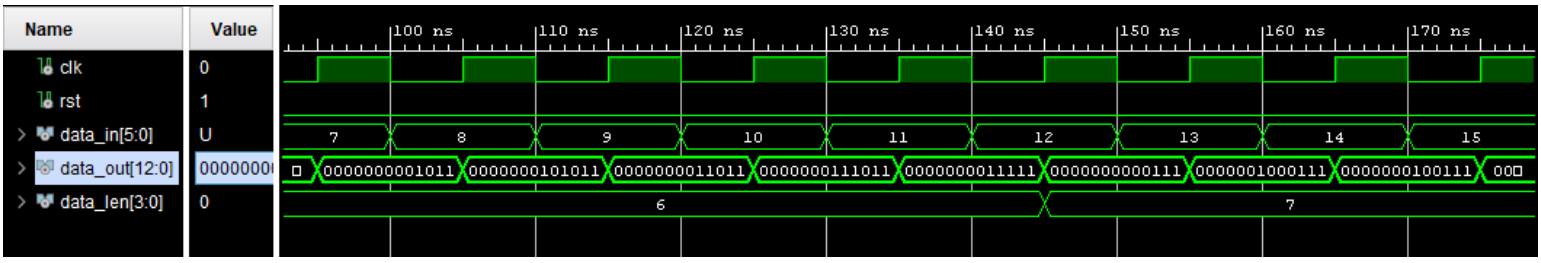


Şekil 4.35 B slice mb type binarization modülüne ait RTL şematik

Resource	Utilization	Available	Utilization %
LUT	28	41000	0.07
FF	21	82000	0.03
IO	25	300	8.33

Şekil 4.36 B slice mb type binarization modülüne ait kaynak kullanım sonuçları

Önceki kısımlarda anlatıldığı gibi simülasyon sonuçlarımız doğru çıktıları verdiği aşağıdaki görselde belirtilmektedir.

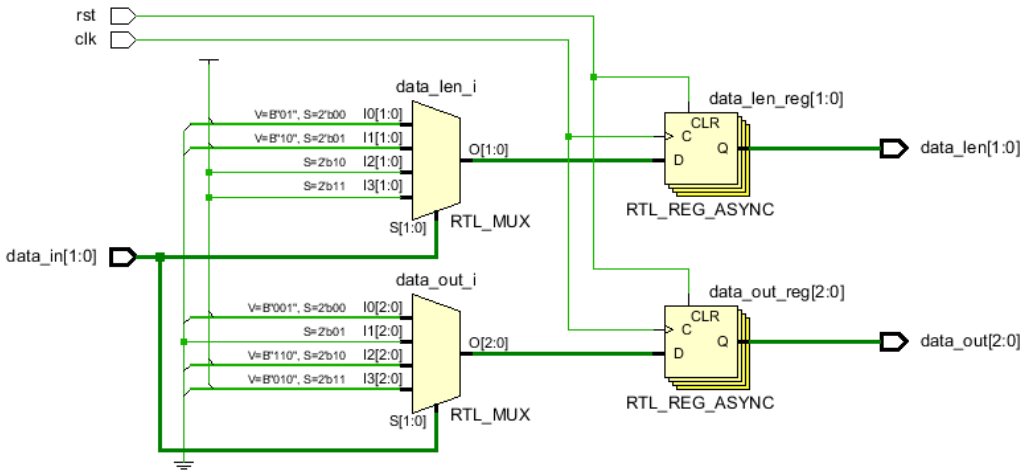


Şekil 4.37 B slice mb type binarization modülüne ait simülasyon sonuçları

4.2.9 B sub mb type binarization

Aşağıda p sub mb type binarization modülüne ait RTL şematik, simülasyon sonucu ve kaynak kullanım sonuçları verilmiştir. Bu sonuçlara göre:

- Kullanılan LUT sayısı:2
- Kullanılan FF sayısı: 5
- Kullanılan I/O portları sayısı: 9

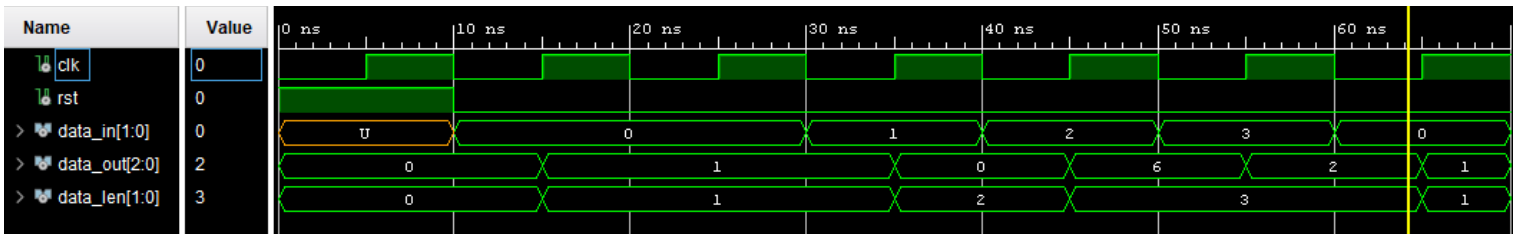


Şekil 4.38 P sub mb type binarization modülüne ait RTL şematik

Resource	Utilization	Available	Utilization %
LUT	2	41000	0.00
FF	5	82000	0.01
IO	9	300	3.00

Şekil 4.39 P sub mb type binarization modülüne ait kaynak kullanım sonuçları

Önceki kısımlarda anlatıldığı gibi simülasyon sonuçlarımız doğru çıktıları verdiği aşağıdaki görselde belirtilmektedir.

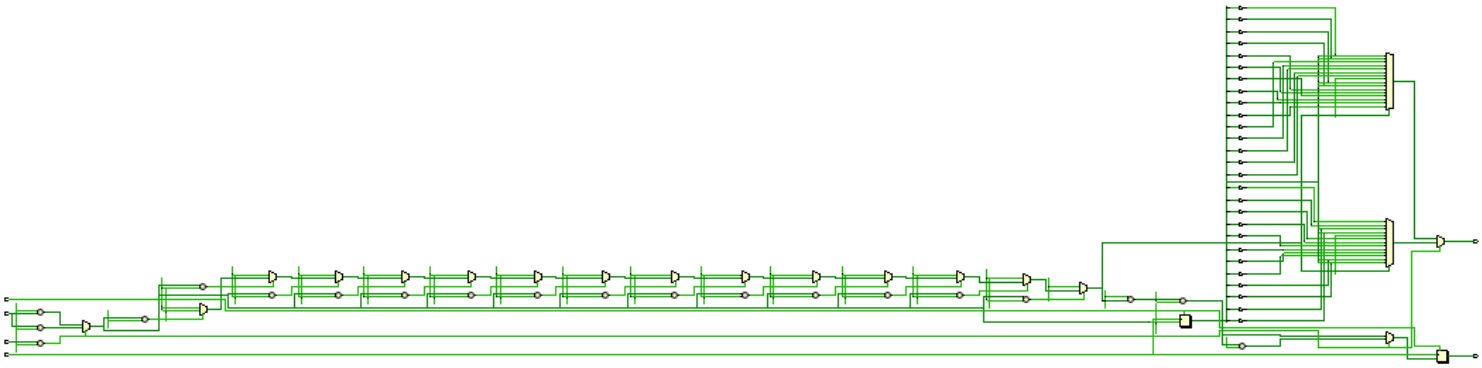


Şekil 4.40 P sub mb type binarization modülüne ait simülasyon sonuçları

4.2.10 Exp golomb binarization

Aşağıda exp golomb binarization modülüne ait RTL şematik, simülasyon sonucu ve kaynak kullanım sonuçları verilmiştir. Bu sonuçlara göre:

- Kullanılan LUT sayısı:162
- Kullanılan FF sayısı: 21
- Kullanılan I/O portları sayısı: 54



Şekil 4.41 Exp golomb binarization modülüne ait RTL şematik

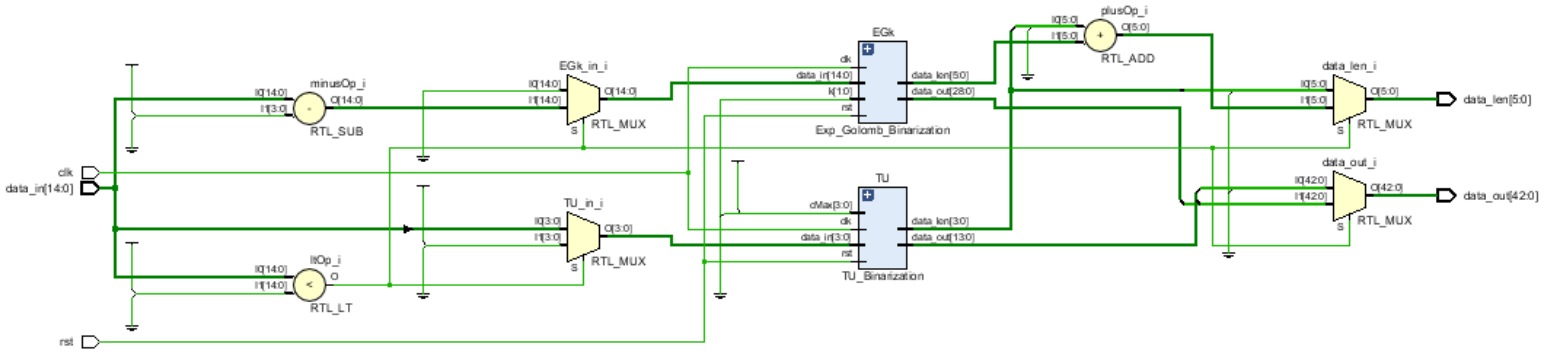
Resource	Utilization	Available	Utilization %
LUT	162	41000	0.40
FF	21	82000	0.03
IO	54	300	18.00

Şekil 4.42 Exp golomb binarization modülüne ait kaynak kullanım sonuçları

4.2.11 UEGK binarization

Aşağıda UEGK binarization modülüne ait RTL şematik, simülasyon sonucu ve kaynak kullanım sonuçları verilmiştir. Bu sonuçlara göre:

- Kullanılan LUT sayısı:146
- Kullanılan FF sayısı: 37
- Kullanılan I/O portları sayısı: 66

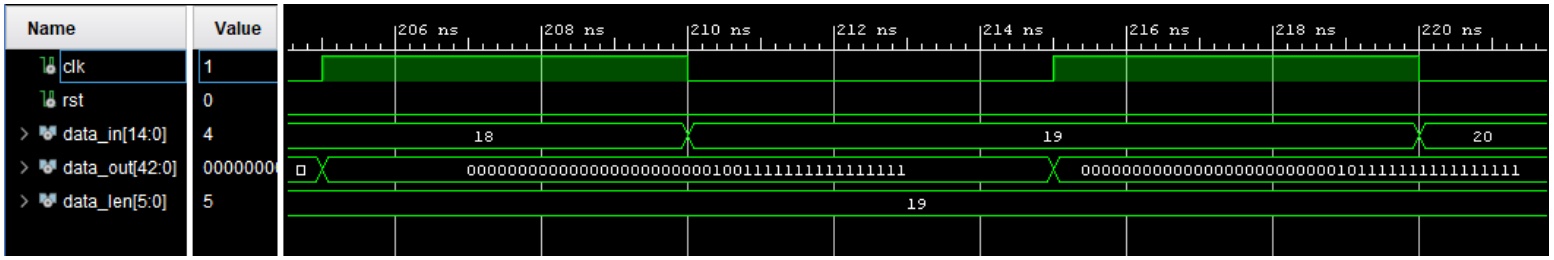


Şekil 4.43 UEGK binarization modülüne ait RTL şematik

Resource	Utilization	Available	Utilization %
LUT	146	41000	0.36
FF	37	82000	0.05
IO	66	300	22.00

Şekil 4.44 UEGK binarization modülüne ait kaynak kullanım sonuçları

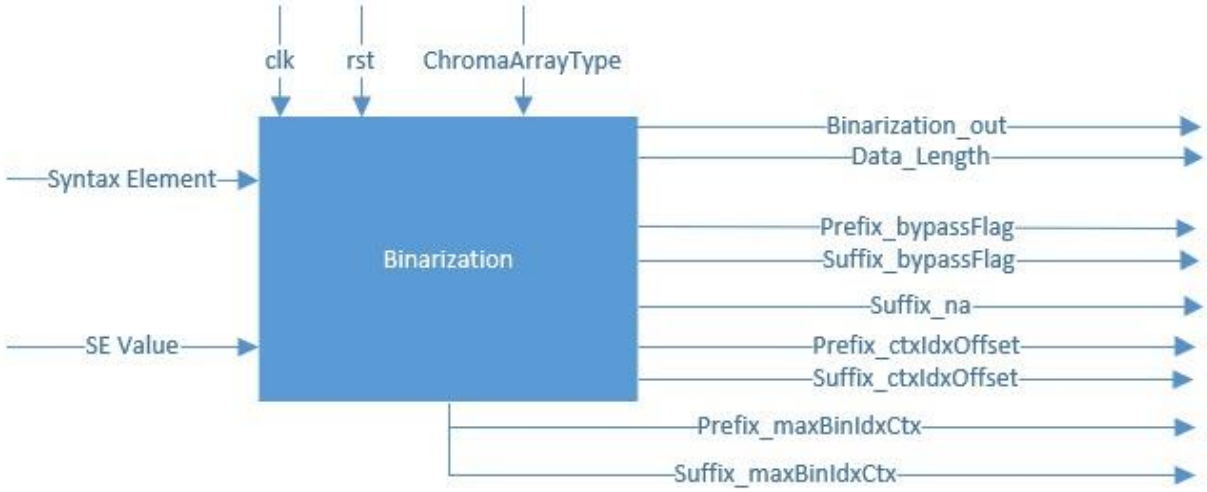
Önceki kısımlarda anlatıldığı gibi simülasyon sonuçlarımız doğru çıktıları verdiği aşağıdaki görselde belirtilmektedir.



Şekil 4.45 UEGK binarization modülüne ait simülasyon sonuçları

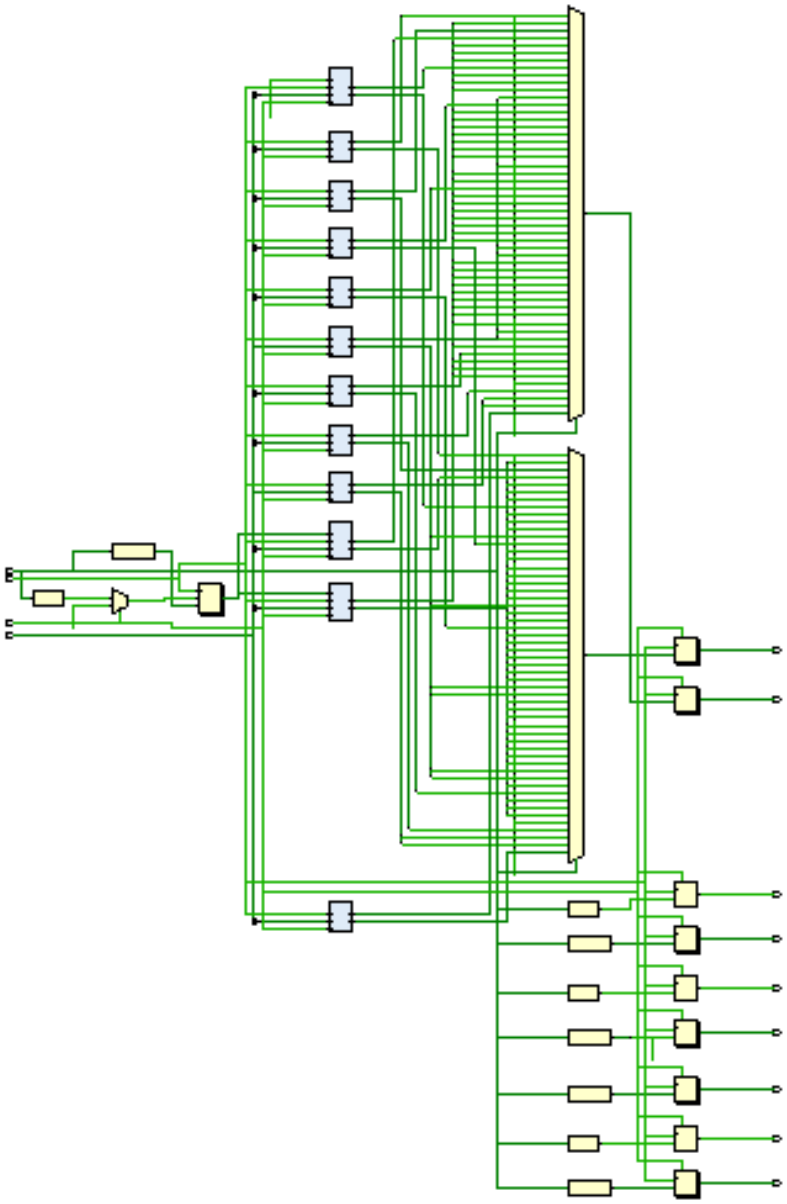
4.2.12 Binarization top level tasarım

Aşağıda tasarımını yaptığımız binarization top modülünün blok diyagramı görülmektedir ayrıca Binarization top modülüne ait RTL şematik, simülasyon sonucu ve kaynak kullanım sonuçları verilmiştir. Bu sonuçlara göre:



Şekil 4.46 Binarization modülünün blok diyagramı

- Kullanılan LUT sayısı:647
- Kullanılan FF sayısı: 351
- Kullanılan I/O portları sayısı: 114



Şekil 4.47 Binarization top modülüne ait RTL şematik

Resource	Utilization	Available	Utilization %
LUT	647	41000	1.58
FF	351	82000	0.43
IO	114	300	38.00

Şekil 4.48 Binarization top modülüne ait kaynak kullanım sonuçları

4.3 Literatür Karşılaştırması

Aşağıda verilen tabloda literatürde bulunan 2 çalışmayla tasarımımızın frekan ve slice sayısı hakkında karşılaştırması verilmiştir. Bu doğrultuda literatürdeki çalışmalar ile slice sayıları arasında ciddi bir fark olduğu ve frekans değerinin ise yaklaşık %40 oranında daha düşük olduğu gözlemlenmiştir

	Slice Sayısı	**Frekans**
[13]	394	267MHz
[14]	212	247.5MHz
Kendi Tasarımımız	647	154MHz

Tablo 4.1 Literatürdeki çalışmalarla karşılaştırma

5. GERÇEKÇİ KISITLAR, SONUÇLAR VE ÖNERİLER

5.1 Çalışmanın Uygulama Alanı

Proje kamerası olan ve video kaydedebilen her cihaza uygulanabilir.
Depolama ve aktarım için gereken sıkıştırma işlemindeki başarılarından dolayı H.264 standardı oldukça yaygın bir biçimde kullanılmaktadır. TV yayıncılığı veya Blu-Ray diskler için de oldukça yaygın olarak kullanılan bir standarttır.

5.2 Gerçekçi Tasarım Kısıtları

5.2.1 Maliyet

Projenin tasarım aşamasında bir maliyeti bulunmamaktadır. Standarda ilişkin dosya ITU sitesinde açık bir şekilde sunulmaktadır. Tasarım aşamasında kullanılan MATLAB ve RTL kodlama için kullanılan Vivado uygulaması öğrenciler için ücretsiz olan uygulamalardır.

5.2.2 Standartlar

Projenin tamamında ITU H.264/AVC standartları takip edilerek ilerlenilmiştir.

5.2.3 Sosyal, çevresel ve ekonomik etki

Her geçen gün video kullanımı, video yayıncılığı artmakta ve önem kazanmaktadır. Özellikle depolama araçlarının ucuzlamasına rağmen veri boyutları da bir o kadar hızlı şekilde artmaktadır. Bu nedenle sıkıştırma algoritmaları da önem kazanmaya devam etmektedir ki uluslararası komiteler de bu yönde çalışmalarını sürdürmektedir. Gözle görülür bir sosyal ve çevresel etki bırakmasa bile bu tarz projelerin ülkemizde tasarlanması ve üretilmesi ekonomi açısından güçlü bir adım olacaktır. Sadece yurtiçinde değil yurtdışında da artan ihtiyaçlar ithalat kapılarını açacak ve ekonomiye katkı sağlayacaktır.

5.2.4 Sağlık ve güvenlik riskleri

Projede herhangi bir sağlık ve güvenlik riski saptanmamıştır.

5.3 Sonuçlar

Video sıkıştırma standartları üzerine bitirme projesi yapılması sayesinde büyük bir bilgi birikimi elde edilmiştir. Son gelinen aşamada başarılı bir şekilde tamamlanmış ve birçok video ile test edilip doğrulanmış MATLAB modeline sahibiz. Bu model üzerinden tasarımını yaptığımız RTL tarafında ise Binarization bloğunu tamamlanmış ve simülasyon kodları yazılmış ve simülasyon sonuçları incelenmiştir. Bu sonuçlar doğrultusunda binarization bloğunun RTL kısmı doğru bir şekilde tamamlanmıştır.

5.4 Geleceğe Yönelik Öneriler

Öncelikle Context Model ve Aritmetik Kodlama bloklarının tamamlanması ilk hedef olmalıdır. Ardından RTL tasarımın performansı açısından yazılan kodlar daha optimize hale getirilmek istenilebilir, geçmiş bölümlerde anlatıldığı gibi hem alan hem de hız açısından literatürdekilerden geride bir sonuç elde edilmiştir ancak uygulamaya göre bu özellikler yeterli görülebilir. Bu adımdan da sonra H.264 standardından sonra yayımlanan 2 standart olan

H.265 ve H.266 için tek Entropi kodlama yöntemi CABAC olmuştur. Eğer sıkıştırma oranlarını artırıp daha hızlı veri aktarımı veya daha düşük depolama kullanılmak istenilirse H.265 veya H.266 standartlarına geçilip bu standartlara uygun bir şekilde CABAC tasarımının güncellenmesi gelecek çalışmaların konusu olabilir.

KAYNAKLAR

[1] Iain E G Richardson, (2012). The H.264 advanced video compression standard.

[2] "CABAC." ***Iphome.hhi.de*****, iphome.hhi.de/marpe/cabac.html. Accessed 28 Nov. 2022.**

[3] Panayides, Andreas & Pattichis, Marios & Pantzaris, Marios & Constantinides, A. & Pattichis, C. (2020). The Battle of the Video Codecs in the Healthcare Domain - A Comparative Performance Evaluation Study Leveraging VVC and AV1. IEEE Access. PP. 1-1. 10.1109/ACCESS.2020.2965325.

[4] Ravi, Aruna & Rao, Kamisetty. (2011). Performance Analysis and Comparison of the Dirac Video Codec with H.264/MPEG-4 Part 10 AVC. International Journal of Wavelets Multiresolution and Information Processing. 9. 635-654. 10.1142/S0219691311004341.

[5] S. K. Kwon, A. Tamhankar and K. R. Rao, "Overview of H.264 / MPEG-4 Part 10" J. Visual Communication and Image Representation, Vol 17, pp.186-216, April 2006.

[6] D. Kumar, P. Shastry and A. Basu, "Overview of the H.264 / AVC", 8th Texas Instruments Developer Conference India, 30 Nov – 1 Dec 2005, Bangalore.

[7] "Advanced video coding for generic audiovisual services," Int.

Telecommun. Union-Telecommun. (ITU-T) and Int. Standards

Org./Int. Electrotech. Comm. (ISO/IEC) JTC 1, Recommendation

H.264 and ISO/IEC 14 496-10 (MPEG-4) AVC, 2003.

[8] Tandon, A., Shastri, R., Murthy, M. Y. B., Sarma, P., Renjith, P., & Rajesh, M. V. (2022). Video streaming in ultra high definition (4K and 8K) on a portable device employing a Versatile Video Coding standard. *Optik*, *271*, 170164. https://doi.org/10.1016/j.ijleo.2022.170164

[9] Thomson, Gavin; Shah, Athar, (2017). Introducing HEIF and HEVC.

[10] Sayood, K. (2000). Introduction to Data Compression / K. Sayood. Morgan Kaufmann Publishers, Inc.

[11] R. Karthikeyan, G. Sainarayanan, and S. N. Deepa, "Perceptual video quality assessment in H.264 video coding standard using objective modeling," SpringerPlus, vol. 3, no. 1. Springer Science and Business Media LLC, Apr. 04, 2014. doi: 10.1186/2193-1801-3-174.

[12] C. Poynton, *Digital Video and HD: Algorithms and Interfaces*. Amsterdam: Elsevier, 2012.

[13] A. B. Hmida, S. Dhahri and A. Zitouni, "A hardware architecture binarizer design for the H.264/ AVC CABAC entropy coding," 2014 International Conference on Electrical Sciences and Technologies in Maghreb (CISTEM), Tunis, Tunisia, 2014, pp. 1-4, doi: 10.1109/CISTEM.2014.7076749.

[14] Andre Luis del Mestre Martins, Vagner Rosa, Sergio Bampi,

"A Low-Cost Hardware Architecture Binarizer Design for the

H.264/ AVC CABAC Entropy Coding", 17th IEEE International

Conference on Electronics, Circuits, and Systems (ICECS),

2010.

ÖZGEÇMİŞ

Ad-Soyad **: Yiğit Bektaş GÜRSOY**

Doğum Tarihi ve Yeri **: 04.06.1999 Eminönü**

E-posta **:** yigitbektasgursoy@hotmail.com

Eğitim

İstanbul Teknik Üniversitesi
Elektronik ve Haberleşme Mühendisliği (2018-2023)

Deneyimler

1. **IGA** (Istanbul Grand Airport)
   Stajyer (08/2021 – 09/2021)

2. **TUBITAK BILGEM**
   Bursiyer (10/2021 – 04/2022)

3. **TUSAŞ**
   Yarı Zamanlı Tasarım Mühendisi (04/2022 – 10/2022)

4. **YONGATEK**
   Yarı Zamanlı Sayısal Tasarım Mühendisi (10/2022 – Hala devam ediyor)

ÖZGEÇMİŞ

Ad-Soyad **: Fatih Enes DOĞAN**

Doğum Tarihi ve Yeri **: 05.12.2000 Üsküdar**

E-posta **:** doganf19@itu.edu.tr

Eğitim

İstanbul Teknik Üniversitesi
Elektronik ve Haberleşme Mühendisliği (2019-2023)

Deneyimler

1. **TUBITAK BILGEM**
   Stajyer(08/2021 – 09/2022)
   Yarı Zamanlı Stajyer (10/2021-05/2022)

2. **İTÜ GSTL**
   Stajyer (06/2022 - 07/2022)

3. **ARÇELİK A.Ş**
   Stajyer (07/2022 - 08/2022)

4. **Yongatek**
   Yarı Zamanlı Proje Stajyeri (07/2022 - Halen)