**TOBB EKONOMİ VE TEKNOLOJİ ÜNİVERSİTESİ**
**FEN BİLİMLERİ ENSTİTÜSÜ**

# YÜKSEK PERFORMANSLI BİTCOİN MADENCİLİĞİ İÇİN SHA256 ÖZET ALGORİTMASININ ENİYİLENMESİ

**YÜKSEK LİSANS TEZİ**

**Erşen BALCISOY**

**Bilgisayar Mühendisliği Anabilim Dalı**

**Tez Danışmanı: Prof. Dr. Kemal BIÇAKCI**

**Nisan 2017**

Fen Bilimleri Enstitüsü Onayı

……..
**Prof. Dr. Osman EROĞUL**
Müdür

Bu tezin Yüksek Lisans derecesinin tüm gereksinimlerini sağladığını onaylarım.

……
**Doç. Dr. Oğuz ERGİN**
Anabilimdalı Başkan V.

TOBB ETÜ, Fen Bilimleri Enstitüsü'nün 131111029 numaralı Yüksek Lisans Öğrencisi **Erşen BALCISOY**'un ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı **"YÜKSEK PERFORMANSLI BİTCOİN MADENCİLİĞİ İÇİN SHA256 ÖZET ALGORİTMASININ ENİYİLENMESİ"** başlıklı tezi **10.04.2017** tarihinde aşağıda imzaları olan jüri tarafından kabul edilmistir.

| | | |
|---|---|---|
| **Tez Danışmanı :** | **Prof. Dr. Kemal BIÇAKCI**<br>TOBB Ekonomi ve Teknoloji Üniversitesi | ..... |
| **Jüri Üyeleri :** | **Prof. Dr. Ali Aydın SELÇUK (Başkan)**<br>TOBB Ekonomi ve Teknoloji Üniversitesi | ..... |
| | **Yrd. Doç. Dr. Mehmet Efe ÖZBEK**<br>Atılım Üniversitesi | ..... |

## TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, alıntı yapılan kaynaklara eksiksiz atıf yapıldığını, referansların tam olarak belirtildiğini ve ayrıca bu tezin TOBB ETÜ Fen Bilimleri Enstitüsü tez yazım kurallarına uygun olarak hazırlandığını bildiririm.

Erşen BALCISOY

# ÖZET


Yüksek Lisans

YÜKSEK PERFORMANSLI BİTCOİN MADENCİLİĞİ İÇİN SHA256 ÖZET ALGORİTMASININ ENİYİLENMESİ

Erşen BALCISOY

TOBB Ekonomi ve Teknoloji Üniveritesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Prof. Dr. Kemal BIÇAKCI

Tarih: Nisan 2017

2008'in başlarından beri, Bitcoin önemli ölçüde kullanıcının ilgisini çekmekte ve bu merkezi olmayan sanal para biriminin popülerliği her geçen gün artmaktadır. Bitcoin tamamen dağıtılmış, eşten eşe bir sistemdir. Bu nedenle merkezi bir sunucu veya kontrol noktası yoktur. Bitcoin, madencilik adı verilen ve zorlu bir sorunun çözümünü arayan bir süreçle oluşturulur. Bitcoin ağında yer alan herhangi bir katılımcı, bu soruna çözüm bulmaya çalışmak için bilgisayarlarının işlemci gücünü kullanarak madenci olarak çalışabilir. Ortalama olarak her 10 dakikada bir, son 10 dakikalık işlemlerin geçerliliğini doğrulayan yeni bir çözüm, ağda bulunan herhangi bir madenci tarafından bulunmakta ve yeni bitcoinler ile ödüllendirilmektedir. Aslında, Bitcoin madenciliği bir merkez bankasının işlevlerini merkezsizleştirmekte ve herhangi bir merkez bankasının ihtiyacını ortadan kaldırmaktadır. Yıllar boyunca madenciler, Bitcoin maden arenasında ayakta kalabilmek için oldukça yüksek özet işlemi yapma gücüne ihtiyaç duymuşlardır. Madencilik aygıtlarının özet üretme hızı ve enerji tüketimi Bitcoin madenciliğinde kazanç elde etmeyi belirleyen en önemli unsurlardır. Bitcoin madenciliği, tamamen belirli yapıdaki girdinin iki kere SHA256 işlemine tabi tutulmasına dayanmaktadır ve bu işlem için birçok araştırmacı donanım tabanlı optimizasyon yapmayı düşünmüştür. Bitcoin madenciliği için spesifik olan

durumlar göz önüne alınarak SHA256 algoritmasının optimizasyonu üzerine çok az araştırma yapılmıştır. Bu çalışmada literatür taraması sonucu elde edilen bazı yöntemler uygulanarak SHA256 algoritmasının hızlanması amaçlanmaktadır. Burada yapılacak olan işlemler genel SHA256 algoritmasında bir hızlandırma yapmamasına rağmen Bitcoin madenciliği için önemli gelişmeler sağlayacaktır. Önerilen iyileştirme metotları Xilinx Virtex-7 FPGA kartında gerçeklenmiştir ve elde edilen kaynak ve güç tüketimi değerlerine göre bu iyileştirme metotlarının uygulanabilir olduğu sonucuna ulaşılmıştır. Tasarlanan sistem sonucunda elde edilen değerlere bakıldığında performans olarak %7'lik bir artış meydana geldiği sonucuna ulaşılmıştır ve bu değere göre de Bitcoin madenciliği için kullanılan iki SHA256 özet fonksiyonunun işlem süresi yaklaşık olarak 1.8611 SHA256 işlem süresine düşmektedir. Bitcoin madenciliğinde en önemli kısıtlayıcı adım olan elektrik tüketiminde de %7'lik bir düşüş olması beklenmektedir. 2016 yılında Bitcoin madenciliği için 400 milyon $'lık elektrik tüketimi yapıldığı kaynaklarda belirtilmektedir ve bu performans artışına göre de 28 milyon $'lık elektrik tasarrufu elde edilebilir. Böylelikle özet üretme hızında bir artış ve enerji tüketiminde bir düşüş sağlanabilmektedir.

**Anahtar Kelimeler:** Bitcoin, Bitcoin madenciliği, SHA256, FPGA, ASIC.

## ABSTRACT

Master of Science

OPTIMIZATION OF SHA256 HASH ALGORITHM FOR HIGH PERFORMANCE BITCOIN MINING

Erşen BALCISOY

TOBB University of Economics and Technology
Institute of Natural and Applied Sciences
Department of Computer Engineering

Supervisor: Prof. Dr. Kemal BIÇAKCI

Date: April 2017

As a decentralized virtual currrency, Bitcoin has attracted many users since 2008. Bitcoin is a fully distributed, peer to peer system. For this reason, there is no central server or point of control. Bitcoins are created through a process called mining, which involves looking for a solution to a difficult problem. Any participant in the Bitcoin network may operate as a miner, using their computer's processing power to attempt to find solutions to this problem. Every 10 minutes on average, a new solution is found by someone who then is able to validate the transactions of the past 10 minutes and is rewarded with new bitcoins. Essentially, Bitcoin mining decentralizes functions of a central bank and replaces the need for any central bank. Miners need high computational power for hash processing. The hashing rate and energy consumption of mining devices are the most important things to earn profit in Bitcoin mining. Bitcoin mining fully relies on performing double SHA256 operation with structured input and many researchers thought about making hardware optimization for this process without considering specifies of Bitcoin mining. Only a few researches have been made on SHA256 hashing algorithm optimization focusing on Bitcoin mining application. In this research, to speed up of SHA256 hashing algorithm, implementing the methods proposed in the recent research is aimed. These

techniques could provide important improvement on Bitcoin mining, but not for general SHA256 hashing algorithm. The proposed optimization methods have been implemented on the Xilinx Virtex-7 FPGA board, and we see that these improvement methods are applicable based on the obtained source and power consumption values. The values collected from the implemented system show us that 7% increase in performance is achieved (the processing time of two SHA256 hash functions used for Bitcoin mining falls to approximately 1.8611 SHA256 time operation). It is expected that around 7% decrease in electricity consumption, which is the most important restriction step in Bitcoin mining, is possible in 2016, $400 million was spent on electricity consumption for Bitcoin mining and according to this performance increase, $28 million electricity consumption can be saved.

**Keywords:** Bitcoin, Bitcoin mining, SHA256, FPGA, ASIC.

## TEŞEKKÜR

Çalışmalarım boyunca değerli yardım ve katkılarıyla beni yönlendiren hocam Prof. Dr. Kemal BIÇAKCI'ya, kıymetli tecrübelerinden faydalandığım TOBB Ekonomi ve Teknoloji Üniversitesi Bilgisayar Mühendisliği Bölümü öğretim üyelerine, eğitimim boyunca bana burs veren TOBB Ekonomi ve Teknoloji Üniversitesine ve destekleriyle her zaman yanımda olan aileme ve arkadaşlarıma çok teşekkür ederim.

**İÇİNDEKİLER**

## ŞEKİL LİSTESİ

**Sayfa**

# ÇİZELGE LİSTESİ

**Sayfa**

## KISALTMALAR

**BTC** **:** Bitcoin
**SHA** **:** Güvenli Özet Algoritması
**FPGA** **:** Field Programmable Gate Array
**LUT** **:** Look-up Table
**MAC** **:** Message Authentication Code
**NIST** **:** National Institute of Standards and Technology

# 1. GİRİŞ

Bitcoin, dijital para ekosisteminin temelini oluşturan teknolojilerin ve kavramların toplamı olarak tanımlanabilmektedir. Bitcoin para birimi, Bitcoin ağında bulunan katılımcılar arasında iletilebilmekte veya biriktirilebilmektedir. Bitcoin kullanıcıları birbiri ile temel olarak İnternet yoluyla Bitcoin protokolü üzerinden iletişime geçebilmektedir. Bitcoin protokol yığını açık kaynak yazılımı olarak geliştirilmiştir ve akıllı telefonlar, bilgisayarlar gibi hesaplama yapabilen tüm cihazlarda doğrudan çalıştırılabilmektedir.

Kullanıcılar, Bitcoin'i mal alıp satmak, insanlara veya kuruluşlara para göndermek gibi geleneksel para birimleriyle yapılabilecek herşeyi yapmak için ağ üzerinden aktarabilirler. Bitcoin teknolojisi, Bitcoin ağının güvenliğini sağlamak için şifrelemeye ve dijital imzalara dayanan özellikleri içerir. Bitcoinler alınabilir, satılabilir ve aynı zamanda diğer para birimlerine de çevrilebilir. Bitcoin'in, İnternet için mükemmel bir para biçimi olduğu düşünülmüktedir [1].

Geleneksel para birimlerinin aksine, Bitcoin tamamen sanaldır. Para olarak, işlemlerde göndericiden alıcıya sadece değer olarak tanımlanmış bir sanal birim gönderilmektedir. Bitcoin kullanıcıları kendilerine ait olan kriptografik anahtar ile Bitcoin ağında yapılan işlemin kendilerine ait olduğunu kanıtlamakta ve işlemden kazandıkları Bitcoin'i harcayabilmekte veya yeni bir alıcıya aktarabilmektedirler. Bu anahtarlar genellikle her kullanıcının bilgisayarında bir dijital cüzdan içine kaydedilir. Bir işlemin kilidini açan anahtara sahip olmak Bitcoin harcamanın tek şartıdır ve böylelikle sistem tarafından kontrol tamamen kullanıcıların eline bırakılmaktadır.

Bitcoin protokolü, ağ üzerindeki madencilik işlevini düzenleyen dahili algoritmaları içerir. Madencilerin çözmesi gereken sorunun zorluğu dinamik olarak ayarlanır, böylece ne kadar çok madencinin (ve merkezi işlemci birimlerinin) sorunun üzerinde çalıştığından bağımsız olarak ortalama 10 dakikada bir doğru cevap bulunur. Protokol aynı zamanda her dört yılda oluşturulan Bitcoin sayısını yarıya düşürecek

şekilde ve toplam miktar olarak 21 milyon Bitcoin'i aşmayacak şekilde sınırlandırılmıştır [1].

Şu andaki pazar değerine bakıldığında 1 BTC (Bitcoin) yaklaşık olarak 1095 $ değerindedir ve 6 yıl öncesinde sadece 0.2 $ değerine sahipti [2]. Bitcoin, insanların onu dijital altın olarak görmesinden dolayı her geçen gün hızlı bir şekilde popülerliğini artırmaktadır [3]. Günde yaklaşık olarak iki yüzbinden fazla Bitcoin işlemi yapılmaktadır [4]. Blok madenciliği için önceleri 50 BTC teşvik verilmekte iken bu değer her bir 210000 bloktan sonra yarıya indirilerek şu an 12.5 BTC = 13687.5 $'a düşmüştür. Bundan dolayı Bitcoin madenciliği insanlar için çok çekici bir potansiyel iş olarak görülmektedir. Birçok madenci yeni bloğu ilk üretmek ve ödülü kazanmak için ciddi yarışlara girmektedir. Bitcoin ağının özet üretme hızı artmasına rağmen Bitcoin üretiminin belirli bir seviyede kalması için zorluk oranı da artmaktadır.

Yukarıda bahsedildiği gibi yeni Bitcoin elde etmek için kullanılan yöntemlerden en önemlisi Bitcoin madenciliği işlemidir. Bitcoin madenciliğinde temel olarak bir iş kanıtı probleminin çözülmesi gerekmektedir. Bu problemin çözümü sonucunda, belirli uzunluktaki blok başlığı değerinin iki defa SHA256 özet algoritmasından geçirilmesi ve elde edilen değerin sistem tarafından sağlanan hedef değerinden küçük olması ve başında belirli miktarda 0 bulunması gereksinimi sağlanmalıdır. Ağda bulunan madenciler arasında oldukça ciddi bir rekabet yaşanmaktadır ve bu çok sayıda işlemin yapılmasında önemli miktarda elektrik tüketimi gerçekleşmektedir. Bundan dolayı, iş kanıtı probleminin çözümünü hızlandırmak ve elektrik tüketimini daha aşağı seviyelere çekmek için bir takım donanımsal iyileştirmeler öneren çalışmalar bulunmaktadır. Ancak Bitcoin blok başlığı değerindeki bazı alanların iş kanıtı probleminin çözümünü bulana kadar sabit olması veya değerinin 0 olmasından dolayı Bitcoin özelinde bir takım iyileştirmelerin mümkün olduğu literatürde sadece bir çalışmada belirtilmiştir ve bu çalışmadaki iyileştirme metotlarının herhangi bir donanımsal ya da yazılımsal uygulamasının mevcut olmadığı saptanmıştır.

Bu tez kapsaında söz konusu iyileştirme yöntemlerinin donanımsal olarak FPGA'de uygulanması gerçekleştirilmiştir. Böylelikle yaptığımız çalışma sonucunda madencilik işleminin daha hızlı bir şekilde yapılmasını ve bu işlem sırasında oluşan elektrik tüketimini önemli ölçüde azaltmayı hedeflemekteyiz. Yaptığımız tezin ana konusu bu iyileştirme metotlarının donanımda uygulanması ve elde edilen sonuçların

performansa ne kadar etkisinin olduğunun ve uygulandığında donanımdaki kaynak tüketiminde nasıl bir değişimin gerçekleştiğini tespit etmektir.

Tez kapsamında, FPGA'de öncelikle standart SHA256 özet algoritması, sonrasında ise İnternette yer alan referans değerlere göre kontrolü yapılarak Bitcoin madenciliği için gerekli olan iki SHA256 özet algoritması ve son olarak da önerilmiş olan iyileştirme yöntemleri gerçekleştirilmiştir. Bu işlemlerin yapılmasında donanım olarak Xilinx Virtex7 FPGA kartı kullanılmıştır. Bölüm 2'de Bitcoin hakkında detaylı bilgiler verilecektir, özellikle ağda bulunan bir kullanıcının sistemde ne tür işlemler yaptığının ve bu işlemlerin hayat döngüsü anlatılacaktır. Bölüm 3'de Bitcoin madenciliğinin temeli olan SHA256 özet algoritmasından bahsedilerek algoritmada yer alan işlemlerin detayları ele alınacaktır. Bölüm 4 ve 5'de Bitcoin madenciliğinden bahsedilerek bu konu hakkındaki önemli terimlere açıklamalar getirilecektir ve önerilmiş olan iyileştirme metotlarından bahsedilecektir. Bölüm 6'da ise FPGA'de gerçekleştirilmiş olan SHA256 özet algoritmasının ve Bitcoin madenciliği için geliştirmiş olduğumuz donanımsal uygulamaların detayları verilerek, sonuç kısmında elde edilen performans artışı ve kaynak tüketiminin durumu incelenecektir. Yapmış olduğumuz çalışmanın kaynak kodları (https://github.com/ebalcisoy/thesis) erişime açıktır.

## 2. BİTCOİN'E GENEL BAKIŞ

Bitcoin, evrensel ve merkezsizleşmiş yani herhangi bir devlet veya diğer hukuki varlıklara bağlı olmayan sanal bir para birimidir [5]. Satoshi Nakomoto, b-para ve HashCash gibi bazı önceki benzer uygulamaları biraraya getirerek tamamen merkezileşmemiş bir elektronik para sistemi oluşturmuştur ve yaptığı çalışmayı 2008 yılında "Bitcoin: Eşten eşe Elektronik Nakit Ödeme Sistemi" başlıklı bir makale ile yayınlamıştır. Yayınlamış olduğu bu makalede Bitcoin'in herhangi bir kuruluşa veya üçüncü tarafa ihtiyaç duymadan tamamen noktadan noktaya çalışan bir elektronik para birimi olduğunu ileri sürmektedir [5]. Önerisine göre tasarlamış olduğu elektronik ödeme sistemi, eski metotlarda yer alan güven mekanizması yerine tam olarak kriptografik kanıt üzerine kurulmuştur. Bu yaklaşım sonucunda iki taraf birbiri ile bir alışveriş yaptığında bu alışverişin doğrulanması için herhangi bir üçüncü tarafa ihtiyaç duyulmamaktadır. Önerilen sistem şu şekilde gerçekleşmektedir. Paranın el değiştirmesi sırasında her kullanıcı parayı bir sonrakine gönderirken kendi dijital imzasıyla bir önceki işlemin özetini ve bir sonraki sahibin açık anahtarını imzalar. Alıcı taraf ise bu imzayı gönderici tarafın açık anahtarı ile doğrulayarak işlemin kendisine ait olup olmadığına karar verebilmektedir.

Bitcoin sistemindeki kilit yenilik, önceki dijital para birimlerinin en önemli zayıflığı olarak göze çarpan çift harcama sorununun çözümündedir. Buradaki sözkonusu problem ödeme alanın zincirdeki önceki sahiplerden birinin parayı iki kez kullanmadığını doğrulayamamasıdır. Yaygın bir çözüm, merkezi bir otoritenin (banka, merkez) her işlemin mükerrer işlem olup olmadığını kontrol etmesidir. Her işlemden sonra para merkeze geri döner ve yerine yeni bir para piyasaya sürülür. Sadece merkez tarafından doğrudan piyasaya sürülen paraların iki kez harcanmadığından emin olabiliriz. Bu çözümdeki sorun para sisteminin tüm kaderinin her işlemin üzerinden geçtiği banka gibi bir merkezi kuruluşun elinde olmasıdır. Ödeme alan kişinin, paranın önceki sahiplerinin önceden işlem imzalamadıklarını doğrulayabileceği bir yönteme ihtiyaç duyulmaktadır. Bir işlemin

gerçekleşmediğini kanıtlamanın tek yolu tüm işlemlerden haberdar olmaktır. Güvenilen bir taraf olmadan bunu başarabilmek için işlemler açıkça ilan edilmelidir. Bunun içinde, Nakamoto'nun önerisi şu şekildedir; Bitcoin sisteminde, bütün para transferleri hangi adresten, ne zaman, ne kadar para gittiğini gösteren herkese açık bir muhasebe defteri (ledger) üzerinde toplanır ve bu şekilde hangi hesapta ne kadar para olduğunun bilgisi tutulur ve aynı paranın birden fazla harcanması engellenir.

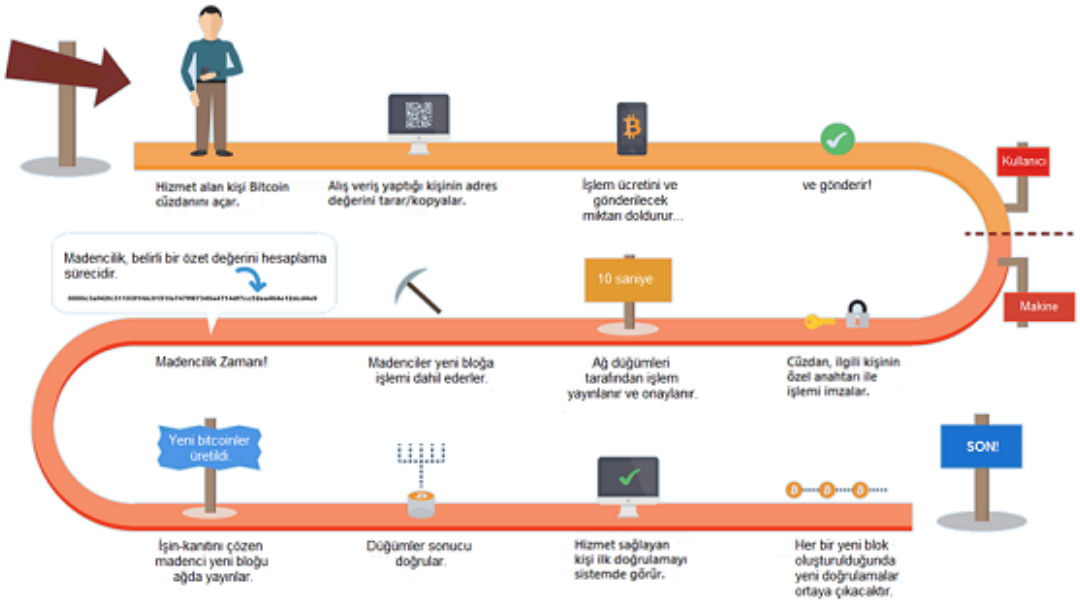
Satoshi Nakamoto'nun icadı, Bizans Generalleri sorunu olarak bilinen, dağıtılmış hesaplamada daha önce çözülmemiş bir probleme yönelik pratik bir çözümdür. Kısacası, güvenilir bir merkezi kurum olmaksızın fikir birliğini sağlamak için İş Kanıtı kavramını kullanan Satoshi Nakamoto'nun çözümü, dağıtılmış sistemler bilimlerinde bir atılım olduğu ve para biriminin ötesinde geniş bir uygulanabilirliğe sahip olduğu düşünülmektedir. Kanıtlanabilir adil seçimler, piyangolar, varlık kayıtları, dijital noter tasdik ve daha fazlası için merkezi olmayan ağlar üzerinde uzlaşmayı sağlamak için kullanılabilmektedir.

Bitcoin ağı, Nakamoto tarafından yayınlanan ve birçok diğer programcı tarafından revize edilen bir referans uygulamasına dayalı olarak 2009'da başlatılmıştır. Bitcoin için güvenlik ve esneklik sağlayan dağıtılmış hesaplama katlanarak arttı ve artık dünyanın en büyük süper bilgisayarlarının işleme kapasitesini aştığı görülmektedir. Bitcoin'in toplam piyasa değerinin dolar/bitcoin kuruna bağlı olarak 5 ila 10 milyar ABD doları arasında olduğu tahmin edilmektedir. Şimdiye kadar sistem tarafından işlenen en büyük işlem 150 milyon dolar olarak gerçekleştirilmiştir ve herhangi bir ücret ödemeden işleme alınmıştır [1].

Bitcoin tamamen dağıtılmış, eşler arası bir sistemdir, fakat bütünlük ve isimsizliğin sağlanabilmesi için kriptografik teknikler de ustalıkla uygulanmıştır. Bu nedenle merkezi bir sunucu veya kontrol noktası yoktur. Bitcoinler, madencilik adı verilen ve zor bir sorunun çözümünü arayan bir süreçle oluşturulmaktadır. Bitcoin ağında yer alan herhangi bir katılımcı (yani Bitcoin protokol yığını kullanan herhangi bir cihaz) bu soruna çözüm bulmaya çalışmak için bilgisayarlarının işlem gücünü kullanarak madenci olarak çalışabilir. Ortalama olarak her 10 dakikada bir Bitcoin ağındaki herhangi bir kullanıcı tarafından son 10 dakikalık işlemlerin geçerliliğini doğrulayan yeni bir çözüm bulunmakta ve bu kullanıcı yeni bitcoinler ile ödüllendirilmektedir. Temel olarak, Bitcoin madenciliği bir merkez bankasının para basma ve dağıtma

işlemini merkezsizleştirir ve herhangi bir merkez bankasının gereksinimini ortadan kaldırır.

Bitcoin ağına yeni katılan bir kullanıcının hangi adımları gerçekleştirmesi gerektiğini şu şekilde açıklayabiliriz. Öncelikle, Bitcoinlerini saklaması için gerekli olan sanal cüzdanını bilgisayarına ya da mobil cihazına kurmalıdır. Bu işlemi gerçekleştirdikten sonra kendisine ait bir adres değerine, herkesin ulaşabileceği açık anahtara ve sadece kendisine ait olan özel anahtara sahip olmalıdır. Bu işlemi tamamladıktan sonra Bitcoin ile bir alım gerçekleştirmek için gerekli olan sanal para dört yolla temin edilebilir. Bunlardan en yaygını Bitcoin kullanan bir tanıdık veya arkadaştan temin edilmesidir, diğer yöntemler ise Bitcoin ATM'lerinden döviz/bitcoin kuruna bağlı olarak satın alınması, kendisine ait bir ürünün veya hizmetin Bitcoin karşılığında satılması ve bazı internet siteleri üzerinden Bitcoin satan kişilere erişerek belli bir ücret karşılığında alınması olarak sıralayabiliriz.




Şekil 2.1 : Bitcoin sisteminde gerçekleşen işlemlerin genel görünümü [6].

Bitcoin ağında yeni bir kullanıcı, yukarıdaki işlemleri yaptıktan sonra sahip olduğu Bitcoin ile bir alış veriş yapmak istediğinde sistemde meydana gelen işlemler zincirini Şekil 2.1'de görüldüğü gibi şu şekilde özetleyebiliriz. Öncelikle ürün veya hizmet alacak kişi bilgisayarında veya mobil cihazında bulunan sanal cüzdanını açar ve aldığı hizmet ve ürüne sahip olan kişinin adresini içeren kare kodu tarayarak ilgili kişinin bilgilerine kolayca ulaşabilmektedir. Bilgilerine ulaştığı kişiye ödemesi gereken miktarı ve Bitcoin madenciliği yapan kişiye verilecek olan işlem ücretini de

ekleyerek sanal para gönderim işlemini tamamlar. Artık kullanıcının yapacağı kısım bitmiştir ve sonrasındaki işlemler Bitcoin sisteminin çalıştığı makinalar ve Bitcoin madencileri tarafından yapılacaktır. Bu aşamada gönderilen sanal para miktarının yanında cüzdan tarafından ilgili kişinin özel anahtarı yardımı ile dijital imza oluşturulur ve bu bilgiler Bitcoin ağındaki tüm kullanıcılara yayınlanır ve onaylanır. Bitcoin ağında bulunan madencilere bu işlem bilgisi geldiğinde yeni Bitcoin ödülü kazanabilmek için birçok işlemi blok adı verilen işlem kümesine dahil ederler. Bundan sonra madenciler iş kanıtı problemini çözmek için sahip oldukları donanımlarda SHA256 özet algoritmalarını koşturmaktadırlar. Herhangi bir madenci iş kanıtı problemini çözdükten sonra işlemlerin yer aldığı yeni bloğu Bitcoin ağında yayınlar. Ağda bulunan düğümler tarafından blok doğrulanır, bu bloğu üreten madenci günümüzde 12.5 BTC ile ödüllendirilmektedir ve yeni bir blok oluşturulurken önceki bloğun özet değeri kullanılır, böylelikle düğümlerin bloğu kabul ettikleri anlaşılmaktadır. Kabul edilen bloktan sonra yeni bir blok üretildiğinde alış veriş sonucunda Bitcoini alacak kişi artık işlemin geçerli olduğunu anlamakta ve sanal paraya karşılık vermesi gereken ürün veya hizmeti alıcıya göndermektedir. Elektronik nakit ödeme sistemi Bitcoinin çalışma prensipleri kısaca bu şekilde gerçekleşmektedir ve her bir işlem için aynı hayat döngüsü tekrar etmektedir.

Bitcoin sisteminde önemli olan terimler aşağıda ele alınmıştır.

**İşlem:** Bitcoin işlemi, temel olarak alıcı ve gönderici arasında gerçekleşen bitcoin transferi olarak adlandırılabilir. Yapılan her bir işlem, bloklarda toplanır ve noktadan noktaya Bitcoin ağında yayınlanmadan önce dijital olarak imzalanır. Bitcoin, işlem esnasında takas edilir ve bu işlemin güvenliği ECDSA imzalama algoritmasına dayanmaktadır [7,8]. Alıcı taraf Bitcoin'in birden fazla kullanılmadığına emin olmak ister, bu problemde yukarıda belirtildiği gibi tüm işlemlerin listesi tüm noktalara iletilerek sağlanmıştır.

**Blok:** Daha önceki bloklarda yer almayan veya bazı Bitcoin işlemlerini içeren bir yapıdır. Yeni bloklar Bitcoin madenciliği ile üretilmektedir ve önceden kabul edilmiş blokların oluşturduğu blok zincirine eklenmektedir. İşlemlerin bulunduğu blok bir kişi tarafından doğrulandığında blok zincirine dahil olmaktadır. Blok zinciri, Bitcoin'in birden fazla kullanılması problemini çözmektedir. Bu arada, bloğun en önemli kısmı Bitcoin madenciliği için çok önemli olan başlıktır. Her bir blok

kendinden önce üretilmiş olan bloğun başlık kısmının özetini içermektedir, böylelikle bir zincir yapısının oluşumu sağlanmaktadır.

**İş İspatı ve En Uzun Zincir:** Bir iş ispatı, belli bir hedefi gerçekleştiren zaman ve maliyet açısından zor elde edilen bir veri parçasıdır. Elde edilen verinin hedefi tutturduğu çok basit yollarla kontrol edilebilir olmak zorundadır. Bir iş ispatı üretimi, çok düşük olasılıklı bir rastsal süreç olabilir, böylece hedefe ulaşmak için ortalama olarak çok sayıda deneme ve yanılma yapılmasını gerektirir. Bitcoin'de iş ispatı, blok başlığı özet değerinin başlangıç kısmında belli sayıda sıfır içermesine kadar blok başlığında yer alan nonce alanının artırılması olarak gerçekleştirilmiştir.

İş ispatı, blok oluşturmak için kullanılır. Her bir bloğun verisine iliştirilmiş iş ispatı, bu bloğun ağ tarafından kabul edilmesi için zorunlu tutulur. Bu işin zorluğu, Bitcoin ağının ortalama 10 dakikada bir blok oluşturmasını sağlayacak şekilde ayarlanır. Başarılı bir blok oluşturmanın olasılığı oldukça düşük olduğundan, ağda yer alan ve bir sonraki blok üzerinde çalışan hangi madencinin bu bloğu oluşturacağı önceden tahmin edilemez.

Bir bloğun geçerli sayılabilmesi için, blok özetinin hedef değerinden küçük olması gerekmektedir. Eğer bu tür blok özet değeri bulunmuş ise iş kanıtı probleminin çözüldüğü ve belli bir miktarda iş yükünün yapıldığı anlaşılmaktadır. Her bir blok kendisinden önce gelen bloğun özetini içerir, böylece her bir blok bir blok zincirini ve hepsi birlikte yüksek oranda bir iş yükünü içerir. Bir bloğu değiştirmek (ancak bir önceki bloğu içeren yeni bir blok oluşturarak mümkündür) kendisinden sonra gelen bütün blokları tekrar oluşturmak ve içerdikleri bütün iş yükünü tekrar yapmakla mümkün olabilir. Bu yöntemle blok zinciri, üzerinde oynama yapılamayacak şekilde koruma altına alınmıştır [5].

**Hedef:** Bitcoin topluluğu tarafından paylaşılan ve yayımlanan 256 bitlik bir tam sayıdır. Bu değer Bitcoin'de iş ispatı probleminin çözümünü bulma zorluğunu belirlemektedir. Bir bloğun kabul edilebilmesi için Bitcoin protokolü tarafından bir temel gereksinim; blok başlığının özet değeri mevcut hedef değerinden küçük olması olarak tanımlanmıştır. Bundan dolayı, hedef değerinin küçülmesi demek Bitcoin madenciliği ile yeni bir blok oluşturulmasının daha da zorlaşması anlamına gelmektedir. Hedef değerine, son 2016 bloğun çıkarılmasından sonra yeni bir değer ataması yapılmaktadır. Her bir blok yaklaşık olarak 10 dakikada bir çıkarılmaktadır

ve böylelikle 2016 bloğun çıkarılması yaklaşık olarak 2 hafta kadar sürmektedir. Bunun sonucunda, her iki haftadan sonra hedef değerinde değişiklik yapılarak işin ispatı probleminin zorlaşması sağlanmaktadır.

# 3. KRİPTOGRAFİK ÖZET FONKSİYONLARI

Elektronik iletişimin, dünyada devrime yol açtığı konusunda kimsenin herhangi bir şüphesi bulunmamaktadır. Dünyada var olan iletişim, kağıtlara yazılan mektupların postaneden gönderilmesinden, anlık olarak haberleşmeyi sağlayan mail, sohbet siteleri veya Facebook, Google+ gibi uygulamalara doğru gelişim göstermiştir. Geleneksel olarak posta yoluyla yapılan pek çok iletişim faaliyeti artık elektronik yöntemlerle yapılmaktadır. Bu etkinlikler arasında belge, resim, ses ve video aktarımı bulunmaktadır.

Taklit etme gibi dolandırıcılık faaliyetlerinden kaçınmak için iletişimin güvenli olması gerekir. Güvenli bir iletişim sağlamak için, bir kurum tarafından transkript gibi oluşturulan belgeler dijital olarak imzalanabilir. Dijital imza birtakım şifreleme ilkelerini kullanmaktadır. Kriptografik özet fonksiyonları, bilgi güvenliğini sağlamak için temel yapı taşı işlevi görmektedir. Kriptografik özet fonksiyonları kendi başına tam bilgi güvenliği sağlamaz, ancak bilgi güvenliğini sağlayan programlarda kritik role sahiptirler. Bu nedenle, bilgi güvenliği projelerinde kriptografik özet fonksiyonları genel güvenliği ve hesaplama verimliliğini önemli ölçüde etkilemektedir.

Kriptografik özet fonksiyonu, herhangi bir uzunluktaki bir girdi verisini sabit uzunluktaki bir çıktıya dönüştüren bir işlem bütünüdür. Kriptografik özet fonksiyonları, bilgisayar programlarında kullanılan sıradan özet fonksiyonlarından biraz farklıdır, ancak basit olması için tezin geri kalan kısmında özet fonksiyonları olarak adlandırılacaktır. Bir özet fonksiyonunun çıktısı belirli özelliklere sahip olmalıdır. Bu özellikler sırası ile ön görüntü direnci (pre-image resistance), ikinci ön görüntü direnci (second pre-image resistance) ve çarpışma direnci (collision resistance) olarak adlandırılmaktadır. Ön görüntü direnci, özet fonksiyonunun tek yönlü bir fonksiyon olması gerektiğini belirtir. Yani, bir saldırganın belirli bir özet değerinden orjinal verileri belirlemesi mümkün olmamalıdır. İkinci ön görüntü direnci, saldırgana verilen herhangi bir mesaj için, verilen mesajdan farklı ve aynı özete sahip başka bir mesaj bulunması zor olmalıdır. Çarpışma direnci, her mesajın

benzersiz bir özet değerine sahip olması ve bir saldırgan tarafından aynı özet değerine sahip iki mesaj bulmasının zor olması demektir.

Matematiksel olarak, bir özet fonksiyonu (H) aşağıdaki gibi tanımlanır:

$$H: \{0,1\}^* \rightarrow \{0,1\}^n$$

Bu gösterimde, $\{0,1\}^*$ herhangi bir uzunluktaki binary değerlere sahip elemanların kümesini belirtirken, $\{0,1\}^n$ uzunluğu n olan binary değerlere sahip olan elemanların kümesini göstermektedir. Böylece, özet fonksiyonu binary değerlere sahip olan herhangi bir uzunluktaki eleman kümesini sabit uzunluktaki eleman kümesine eşlemektedir. Benzer şekilde, bir özet fonksiyonunun özellikleri aşağıdaki gibi tanımlanabilir:

$x \in \{0,1\}^*$; $\mathbf{y} \in \{0,1\}^n$

1- Ön görüntü direnci: y = H(x) verildiğinde, x değerini bulmak zor olmalıdır.
2- İkinci ön görüntü direnci: x değeri verildiğinde H(x) = H(x') olacak şekilde x'den farklı olarak bir x' değeri bulunmamalıdır.
3- Çarpışma direnci: birbirinden farklı olan x ve x' değerleri için H(x) = H(x') olacak şekilde bir x, x' çiftinin bulunabilirliği oldukça zor olmalıdır.

İkinci ön görüntü direnci ve çarpışma direncinin özellikleri benzer görünebilir ancak fark, ikinci ön görüntü direnci durumunda saldırgana bir mesaj (x) verilir, ancak çarpışma direnci için herhangi bir mesaj verilmez, aynı özet değerini veren herhangi bir iki iletiyi bulmak saldırgana bırakılmıştır. "Zor" veya "bulmak zor" ifadesi ile bilgisayarın bu işlemi gerçekleştirmesi için uzun zamana ihtiyacının olduğu ve çok miktarda bellek gereksiniminin olduğu ima edilmektedir. Örneğin, bir mesajın özet değerinden hesaplanabilmesi için bugünün teknoloji standartlarına sahip bir bilgisayar için uzun yıllar ve oldukça yüksek miktarda bellek gerekmektedir. Dolayısıyla, yapılan hesaplamanın uygulanamaz olduğu kabul edilmektedir. Bilgisayarların işlem gücü, on yıllar boyunca arttıkça, önceden güvenli olarak kabul edilen (ön görüntü, ikinci ön görüntü ve çarpışma direncinin tüm özelliklerine sahip olan) bazı özet fonksiyonlarının artık kırılmış olduğu kabul edilmektedir. Hesaplama gücü arttıkça ve özet fonksiyonlarının kripto analizi gerçekleştirildiğinde, bazı özet

fonksiyonu standartları da zayıf olduğu için revize edilmiştir. Güvenli ve hızlı bir şekilde çıktı üreten özet fonksiyonlarına sahip olmak istenmektedir.

Uygulamada, herhangi bir uzunluktaki mesaj özet fonksiyonuna girmemektedir, maksimum değeri vardır. Bununla birlikte, elde edilen özet değerinin uzunluğuyla karşılaştırıldığında, girdi mesajının uzunluğu rastgele sayılabilir. Bir özet fonksiyonunda, gelen mesaj sabit bir boyuttaki bloklara bölünür (tüm ileti blok boyutundan küçük veya eşit olmadığı sürece). Bu parçalama sürecinde, ileti herhangi bir kalan veya artık bitler olmaksızın çok sayıda bloğa sığması için genellikle genişletilir. Blok boyutu (yani, bir blok içinde bulunan bit sayısı) özet fonksiyonuna bağlıdır. Bir iletiyi genişletmek, iletinin sonuna belirli sayıda sıfır ve biri ekleyerek oluşmaktadır. Genişletme işleminde, aynı zamanda mesaj uzunluğunun değeri genişletme bitleri içerisine gömülmektedir. Özet fonksiyonuna bağlı olarak genişletme bitlerine başka bilgiler de gömülebilmektedir. Genişletme işlemi gerçekleştirilmiş ve bloklara ayrılmış mesaj üzerinde özet fonksiyonu yinelemeli bir şekilde çalışmaktadır. İlk mesaj bloğu özet fonksiyonuna girilir ve ilgili özet değeri elde edilir. İkinci mesaj bloğu daha sonra birinci bloğun özet değeri ile birlikte girilir. Böylece, ilk mesaj bloğundan gelen özet değere bir zincir veya ara değer adı verilmektedir ve ikinci mesaj bloğunun sıkıştırılması için başlangıç değeri olarak özet fonksiyonun girişine geri beslenir. İlk mesaj bloğunun sıkıştırılması için kullanılan başlangıç değeri, belirli bir özet fonksiyonu için sabittir ve sıklıkla başlatma vektörü olarak adlandırılır. Son mesaj bloğu sıkıştırıldığında elde edilen özet değer, daha sonra bloklara bölünmüş olan mesajın özeti olarak adlandırılır [9].

### 3.1 Özet Fonksiyonlarının Uygulamaları

Daha önce belirtildiği gibi, özet fonksiyonları bilgi güvenliği uygulamalarında yaygın olarak kullanılmaktadır. Bunlar arasında dijital imzalar, mesaj doğrulama kodları (MAC) gibi uygulamalar yer almaktadır. Parola depolama gibi özet fonksiyonlarının basit uygulamaları da bulunmaktadır. Parola depolama uygulamasında, ilk girişte bir kullanıcı tarafından girilen parola bilgisayar sisteminde depolanmaz, bunun yerine parolanın özeti depolanmaktadır. Daha sonraki zamanlarda sisteme giriş yapmak için kullanıcının parolayı girmesi gerekir, sistem parolanın özetini elde eder ve onu depolanmış özet değeri ile karşılaştırır. Bir eşleşme varsa sisteme erişim için izin verilir aksi takdirde kullanıcının erişimi kesilir.

Bu tür bir uygulamanın avantajı, eğer bir saldırgan sistemin depolama aygıtlarına erişmeyi başarırsa sadece parolaların özet değerlerine ulaşabilir ve özet fonksiyonunun tek yönlü olmasından dolayı esas parola değerlerini elde etmesi önlenmiş olur.

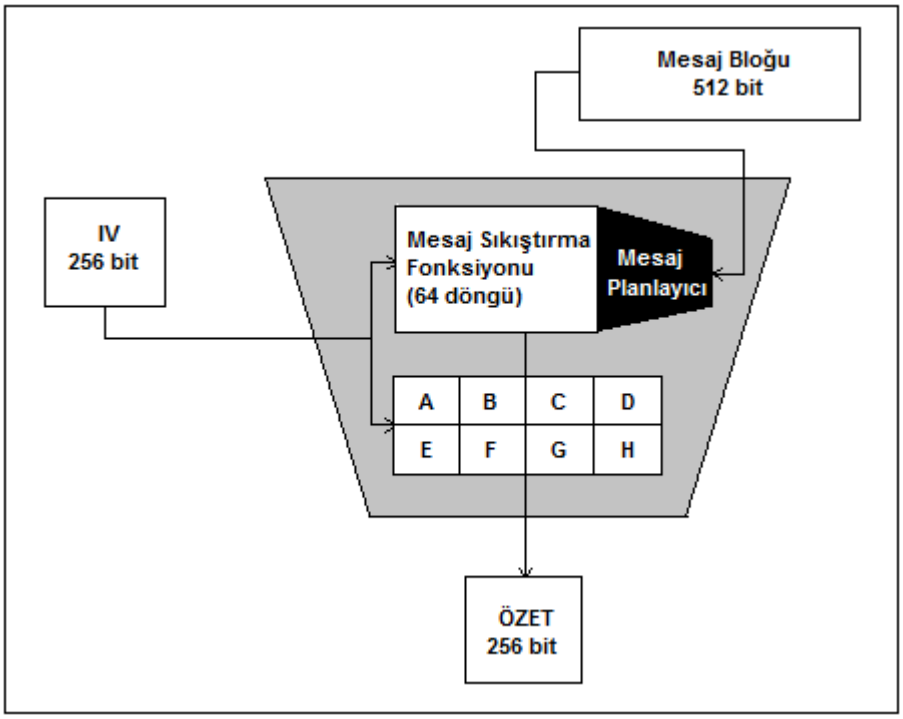
Dijital imzalarda özet fonksiyonları hız bakımından verimliliği artırmak ve uygulamanın bant genişliğini azaltmak için kullanılmaktadır. Dijital imza, bir mesajın özgünlüğünü göstermek için bir araç sağlamaktadır ve şifreleme/şifre çözme gibi asimetrik anahtarların kullanılmasına dayanmaktadır. A tarafı bir mesajı özel anahtarla şifreliyorsa, o mesajın şifresini çözebilen tek anahtar A'nın açık anahtarıdır. Tersine, bir mesaj A'nın genel anahtarı ile başarılı bir şekilde çözülürse, mesajın yalnızca A'nın gizli anahtarıyla şifrelenmiş olabileceği veya başka bir deyişle A'dan gelen mesaj olduğu sonucuna ulaşılabilmektedir. A'nın özel anahtarı ile mesajın şifrelenmesi işlemi mesajın imzalanması olarak adlandırılmaktadır. Bununla birlikte, pratikte mesajın kendisi imzalanmaz, mesajın özeti hesaplanır ve özet değeri imzalanır. Bunu yapmanın avantajı, özet değerinin sabit bir uzunluğa sahip olmasından dolayı asimetrik şifreleme ünitesine beslenen girdi, orijinal mesajın boyutundan bağımsız olarak sabit bir uzunluğa sahiptir. Dolayısıyla girdinin bant genişliği sabitlenmiştir ve asimetrik bir anahtarla şifreleme süreci hesaplama açısından yoğun bir işlemdir ve böylelikle girdi boyutu düşürülerek şifreleme işlemi daha hızlı bir şekilde gerçekleştirilmektedir.

Alınan bir mesajın gönderilen mesajla aynı olduğunu doğrulamak için bir mesaj doğrulama kodu (MAC) kullanılabilir. Başka bir deyişle, bir iletinin bozulduğunu veya aktarım sırasında değiştirilmediğini doğrulamak için kullanılabilir. Bununla birlikte, üçüncü taraf gönderenin kimliğini doğrulamak için bu tür bir işlemi kullanamaz, çünkü MAC gönderen ya da alıcı tarafından hesaplanabilir. MAC, bir anahtarlı özet işlemi ile hesaplanır. Bir anahtarlı özet işlemi, girişinde mesaja ek olarak bir anahtar da içerir. Yapılan işlemin başarılı olması anahtarın gizliliğine bağlıdır. Gönderen ve alıcı aynı anahtarı kullanmalı, ancak anahtarı üçüncü taraflardan gizli tutmalıdır. Bir MAC oluşturmak için, gönderen mesajı ve anahtarı, HMAC veya UMAC gibi bir algoritma tarafından belirtilen şekilde özet fonksiyonuna beslemelidir. Oluşturulan MAC ve mesaj daha sonra alıcıya gönderilir. Mesajın değiştirilmediğini doğrulamak için, alıcı mesaj girdisini aynı algoritmaya sahip olan özet fonksiyonuna besleyerek MAC değeri üretir ve alıcı tarafından

üretilen MAC, gönderenden alınan MAC ile aynı ise alıcı gelen mesajın herhangi bir nedenden dolayı değişmediği sonucuna ulaşmaktadır. Bu tür bir uygulama bir mesajın bütünlüğünü doğrulamak için kolay bir yol sağlar [9].

### 3.2 SHA256 Özet Fonksiyonunun Genel Yapısı

SHA256 özet algoritmasının ayrıntılı bir açıklaması resmi NIST standardında bulunabilir [10]. Şekil 3.1'de SHA256 özet algoritmasının genel bir görünümü verilmiştir. Bu bölüm, Bitcoin ekosisteminin omurgasını oluşturan SHA256 algoritmasına genel bir bakış sunar. Bitcoin işlemlerinin bütünlüğü, çarpışma direncine ve SHA256 özet algoritmasının ön görüntü direncine bağlıdır. Bitcoin protokolünde SHA256 özet algoritmasınıın iki defa hesaplandığını unutmamak önemlidir.




Şekil 3.1 : SHA256 özet algoritmasının genel görünümü.

16 adet 32 bit kelime dizisi olarak temsil edilen 512 bitlik bir blok boyutu vardır. Bu 512 bitlik blok, mesaj planlayıcısı yardımıyla 32 bitlik kelimelere (Wt) dönüştürülerek sıkıştırma fonksiyonuna beslenmektedir. Bunların her ikisi de daha sonra ayrıntılı olarak açıklanacaktır. Mesaj planlayıcı, 512 bit mesaj bloğunu altmış dört adet 32 bitlik kelime kümelerine genişletir. SHA256 özet algoritması içerisindeki işlemler, 32 bit uzunluğuna sahip A, B, C, D, E, F, G ve H sekiz çalışma

değişkeni üzerinde gerçekleştirilir. Dolayısıyla, SHA256 özet algoritmasının kelime uzunluğu 32 bittir. Bu çalışma değişkenlerinin değerleri her turda hesaplanır ve bu süreç 64 döngü tamamlanıncaya kadar devam etmektedir. Önemli noktalardan birisi, SHA256 özet algoritmasındaki tüm eklemelerin $2^{32}$ modunda gerçekleştirildiğine dikkat edilmelidir.

SHA256, ilk mesaj bloğu için sabitlenmiş 256 bitlik başlatma vektörü (IV) kullanmaktadır. İlk 64 döngü tamamlandıktan sonra elde edilen değer bir sonraki mesaj bloğunun özet değeri hesaplanırken başlatma vektörü olarak işleme dahil edilir, bundan dolayı ilk özet değeri ara değer olarak adlandırılmaktadır. Davies-Meyer yapımı kullanılarak inşa edilmiştir ve 64 döngü sonunda elde edilen çıktı değerine başlatma vektörü eklenmiştir. Böylece, mesajın sıkıştırılması işlevinin 64 döngüsü tamamlandıktan ve başlatma vektörünün eklenmesinden sonra algoritma tarafından 256 bitlik bir ara mesaj özeti üretilmiştir. Tüm mesaj bloklarının özet değerleri alındıktan sonra, girilen mesajın son mesaj özetini oluşturan 256 bitlik bir değer elde edilir. Bitcoin protokolünde, saldırganların öngörülen sıfır sayısıyla başlayan yeni bir blok oluşturmalarında kısayol bulmalarını zorlaştıran algoritma SHA256 özet fonksiyonudur, çünkü SHA256 fonksiyonu çığ özelliğine sahiptir, yani girdilerde meydana gelebilecek küçük bir değişiklik çıktıda tahmin edilemez derecede bir değişime neden olmaktadır. Bir sonraki bölümde SHA256 özet algoritmasında var olan fonksiyonların detaylarından bahsedilecektir.

### 3.3 SHA256 Özet Fonksiyonunun Detayları

SHA256 özet algoritması işlemi, üç farklı işleme bölünebilir. Bunlar aşağıda sırasıyla şu şekildedir.

- **Ön işleme:** özet algoritması tarafından beklenen 512 bitlik bloğun eksik kısımları 0 ile genişletilmektedir ve 512 bitten daha uzun olan mesajlar, 512 bitlik bloklara bölünerek fonksiyona beslenmektedir.
- **Mesaj planlayıcı:** 16 kelimelik giriş mesaj bloğundan altmış dört adet kelime üreten fonksiyondur.
- **Sıkıştırma fonksiyonu:** her döngüde mesaj planlayıcıdan gelen mesaj bağımlı kelimenin gerçek özet işlevinin yapıldığı fonksiyondur.

**- SHA256 ön işleme**

SHA256 ön işleme, mesaj planlayıcısı ve sıkıştırma fonksiyonu uygulanmadan önce yapılması gereken ilk adımdır. Ön işleme safhasında aşağıdaki üç görev sırasıyla yerine getirilir:

- Mesaj, 512 bitin katı olacak şekilde genişletme işlevine tabi tutulur,
- 512 bitten daha uzun olan mesajlar, 512 bitlik mesaj bloklarına bölünür,
- Başlangıç özet değeri ayarlanır.

**- Mesajların 0 ile genişletilmesi**

Önce özeti alınacak mesajın genişletilmesi gerekmektedir. Özeti alınacak mesajın, SHA256'nın blok boyutu olan 512 değerinin tam katı haline getirilmesi için genişletme işlemi yapılmaktadır. Şimdi mesaj uzunluğunun 1 bit olduğunu düşünürsek, mesajın sonuna bir bit 1 eklenir ve öncesine de k adet 0 eklenerek mesajın 512 değerinin tam katı olması sağlanır. Burada, Eşitlik (3.1)'deki denklemin negatif olmayan en küçük çözümüdür [10]:

$$l + 1 + k = 448 \bmod 512 \tag{3.1}$$

Buradan genişletme için kaç bitin gerekli olduğu bulunabilir. Böylece özet fonksiyonunun girdisi elde edilir.

**- Genişletilmiş mesajın bloklara bölünmesi**

Mesaj yukarıda açıklanan mantıkla genişletildikten sonra, mesaj planlamasının ve özet hesaplamalarının başlatılabilmesi için mesaj, N adet 512 bitlik bloklara ayrıştırılır.

**- Başlangıç özet değerinin ayarlanması**

Özet değerini hesaplamaya başlamadan önce, Şekil 3.2'de belirtilen 32 bitlik kelimeler başlangıç özet değeri olarak ayarlanır.

| $H_0^0$ | $H_1^0$ | $H_2^0$ | $H_3^0$ | $H_4^0$ | $H_5^0$ | $H_6^0$ | $H_7^0$ |
|---|---|---|---|---|---|---|---|
| 0x6a09e667 | 0xbb67ae85 | 0x3c6ef372 | 0xa54ff53a | 0x510e527f | 0x9b05688c | 0x1f83d9ab | 0x5be0cd19 |

Şekil 3.2 : Başlangıç özet değerlerinin görünümü.

Bu 8 kelime değerinin çıkış noktasını bilmek ilginçtir. İlk 8 asal sayının kare köklerinin kesirli kısımlarının ilk 32 biti alınarak elde edilmektedir. Bu ilk özet

değeri, önceki bölümde açıklandığı gibi SHA256 algoritması için başlatma vektörü olarak görev yapmaktadır.

**- SHA256 mesaj planlayıcısı**

Ön işlem aşamasının tamamlanmasından sonra mesaj planlayıcı bloğu, ilk 512 bitlik mesaj bloğunu alır ve mesaja bağlı olan $W_t$ kelimelerini çıktı olarak vermektedir. Her tur için mesaj planlayıcısı tarafından çıkarılan 32 bitlik mesaja bağlı kelimeler $W_0$, $W_1$, .. , $W_{63}$ olarak etiketlenmiştir ve bu kelimeler aşağıdaki gibi hesaplanır:

$$\text{For } 0 \leq t \leq 15$$

$$W_t = M_t \tag{3.2}$$

$$\text{For } 16 \leq t \leq 63$$

$$Wt = \sigma_1(W_{t-2}) + W_{t-7} + \sigma_0(W_{t-5}) + W_{t-16} \tag{3.3}$$

Burada $\sigma_0$ ve $\sigma_1$, 32 bitlik kelimeler üzerinde çalışan SHA256 mesaj planlayıcısına özgü iki mantıksal işlevdir. Bu işlevlerin ayrıntıları aşağıda verilmiştir.

$$\sigma_0(x) = ROTR^7(x) \oplus ROTR^{18}(x) \oplus SHR^3(x) \tag{3.4}$$

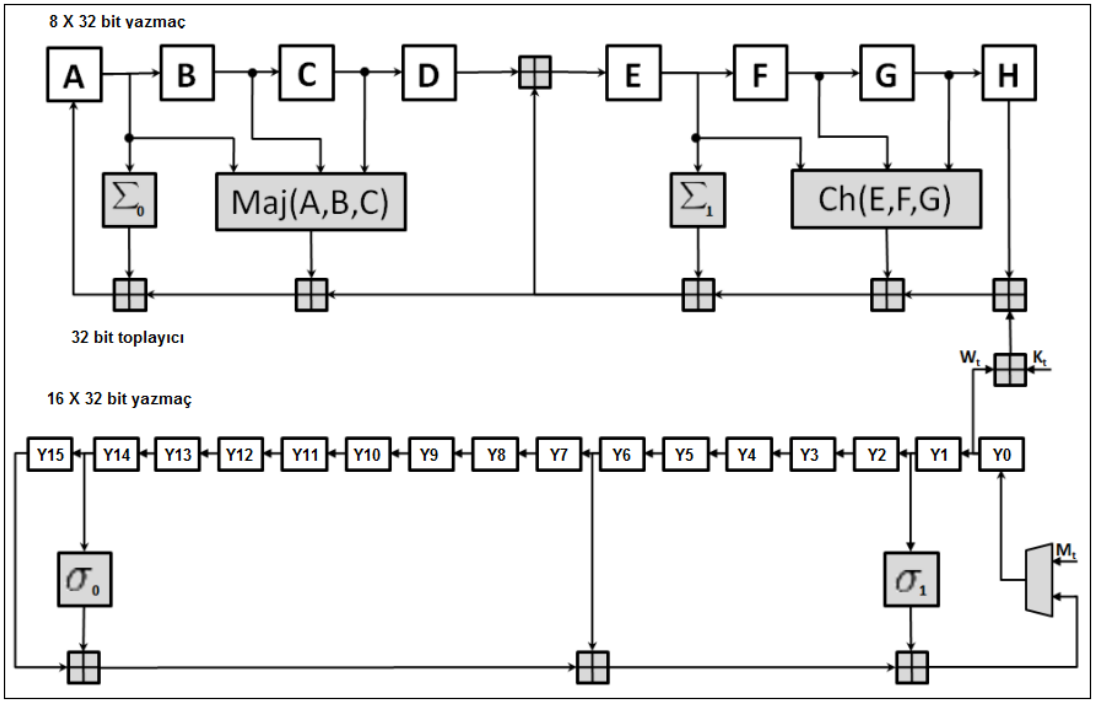
$$\sigma_1(x) = ROTR^{17}(x) \oplus ROTR^{19}(x) \oplus SHR^{10}(x) \tag{3.5}$$

İki mantıksal işlev $\sigma_0$ ve $\sigma_1$, giriş mesajının bir kelimesi üzerinde çalışır ve ona yukarıdaki bit düzeyindeki işlemleri uygulamaktadır. Burada belirtilen $ROTR^x$ fonksiyonu gelen mesajın bit düzeyinde x kadar sağa döndürülmesini gerçekleştirmektedir, $SHR^x$ fonksiyonu ise gelen mesajın bit düzeyinde x kadar sağa kaydırılmasını gerçekleştirmektedir. Son olarak, $\oplus$ simgesi ile gösterilen işlev de bit düzeyinde özel veya mantıksal işlemini gerçekleştirmektedir.

Mesaj planlayıcısının görünümü Şekil 3.3'de verilmiştir. Sistemde bulunan çoklayıcı yardımı ile t döngü sayısına bağlı olarak mesaj değerinin mi yoksa hesaplanmış olan $W_t$ değerinin mesaj planlayıcısına besleneceği kontrol edilmektedir. Her bir döngüde, 32 bitlik $W_t$ değeri kaydırma yazmacı kullanılarak sol tarafa kaydırılmaktadır, bu işlemin yapılmasının amacı $W_t$'nin yeni değerlerinin bulunmasında önceki değerlerin kullanılmasının gerekliliğidir. Açıklanan iki mantıksal işlev $\sigma_0$ ve $\sigma_1$'in, mesaj planlama işleminin 17. döngüsüne kadar fonksiyonda herhangi bir kullanımı söz konusu değildir. 512 bitlik giriş mesajı olduğu gibi ilk 16 döngüde mesaj sıkıştırma fonksiyonuna beslenmektedir.

**-SHA256 mesaj sıkıştırma fonksiyonu**

Mesaj sıkıştırma fonksiyonu, özet fonksiyonunun temel yapısını oluşturmaktadır ve SHA256'nın tek yönlü özelliğini yerine getiren ana işlemdir. Her turda kullanılan ve güncellenen A, B, C, D, E, F, G ve H çalışma değişkenlerinin dışında, iki geçici kelime olan $T_1$ ve $T_2$ mesaj sıkıştırma fonksiyonunda her bir döngüde A ve E değerlerinin hesaplanmasında kullanılmaktadır. Mesaj sıkıştırma fonksiyonunun gerçekleştirdiği ilk adım, bu 8 adet çalışma değişkeninin eğer ilk bloğun özet değeri hesaplanıyorsa başlatma vektörü ile ya da ilk bloktan farklı olarak daha sonraki blokların özet değeri hesaplanıyorsa bir önceki bloğun ara özet değeri ile başlangıç değerlerini ayarlamaktadır.




Şekil 3.3 : SHA256 mesaj sıkıştırma fonksiyonunun ve mesaj planlayıcısının görünümü.

Şekil 3.3'de mesaj sıkıştırma fonksiyonunun tipik uygulanışının yanı sıra, aynı anda çalışan mesaj planlayıcısının da çalışma prensibi görülmektedir. Şekil 3.3'de her turda sekiz çalışma değerinden altısı A, B, C ve E, F, G değerleri bir pozisyon kaydırılarak B, C, D ve F, G, H'ye beslendiği açıkca görülmektedir. $W_t$, 32 bitlik verisi mesaj planlayıcısında hesaplanmaktadır ve sıkıştırma fonksiyonuna beslenmektedir. Her bir döngüye özel şekilde 64 adet $K_t$ adı verilen sabitler bulunmaktadır ve bu sabitler ilk 64 asal sayının küp kökünün sonucunda elde edilen değerin kesirli kısmındaki ilk 32 bitten oluşmaktadır. A ve E değişkenleri tüm girdi

değerlerine bağımlıdır ve ileride anlatılacak olan eşitliklerin kullanımı ile her döngüde hesaplanmaktadır. Daha önce açıklandığı gibi 8 çalışma değişkenine başlangıç değerleri atandıktan sonra, 64 döngü boyunca sıkıştırma fonksiyonu uygulanmaktadır. Uygulanan sıkıştırma fonksiyonu aşağıda belirtilen işlemlerden oluşmaktadır.

$$T_1 = H + \Sigma_1(E) + Ch(E, F, G) + K_t + W_t \quad (3.6)$$

$$T_2 = \Sigma_0(A) + Maj(A, B, C) \quad (3.7)$$

$$H = G;\ G = F;\ F = E \quad (3.8)$$

$$E = D + T_1 = D + H + \Sigma_1(E) + Ch(E, F, G) + K_t + W_t \quad (3.9)$$

$$D = C;\ C = B;\ B = A \quad (3.10)$$

$$A = T_1 + T_2 \quad (3.11)$$

Eşitlik (3.6-3.11)'de bulunan dört mantıksal fonksiyon her bir döngüye beslenen 32 bit uzunluğunda olan $W_t$ kelimesinin karıştırılmasına ve difüzyonuna neden olmaktadır. İlgili eşitlikler 64 döngü boyunca çalışma değişkenlerine uygulandığında çığ etkisi için uygun bir seviye gözlemlenir. Bu dört mantıksal işlevin içerdiği işlemler şu şekildedir.

$$Ch(X, Y, Z) = (X \wedge Y) \oplus (\neg X \wedge Z) \quad (3.12)$$

$$Maj(X, Y, Z) = (X \wedge Y) \oplus (X \wedge Z) \oplus (Y \wedge Z) \quad (3.13)$$

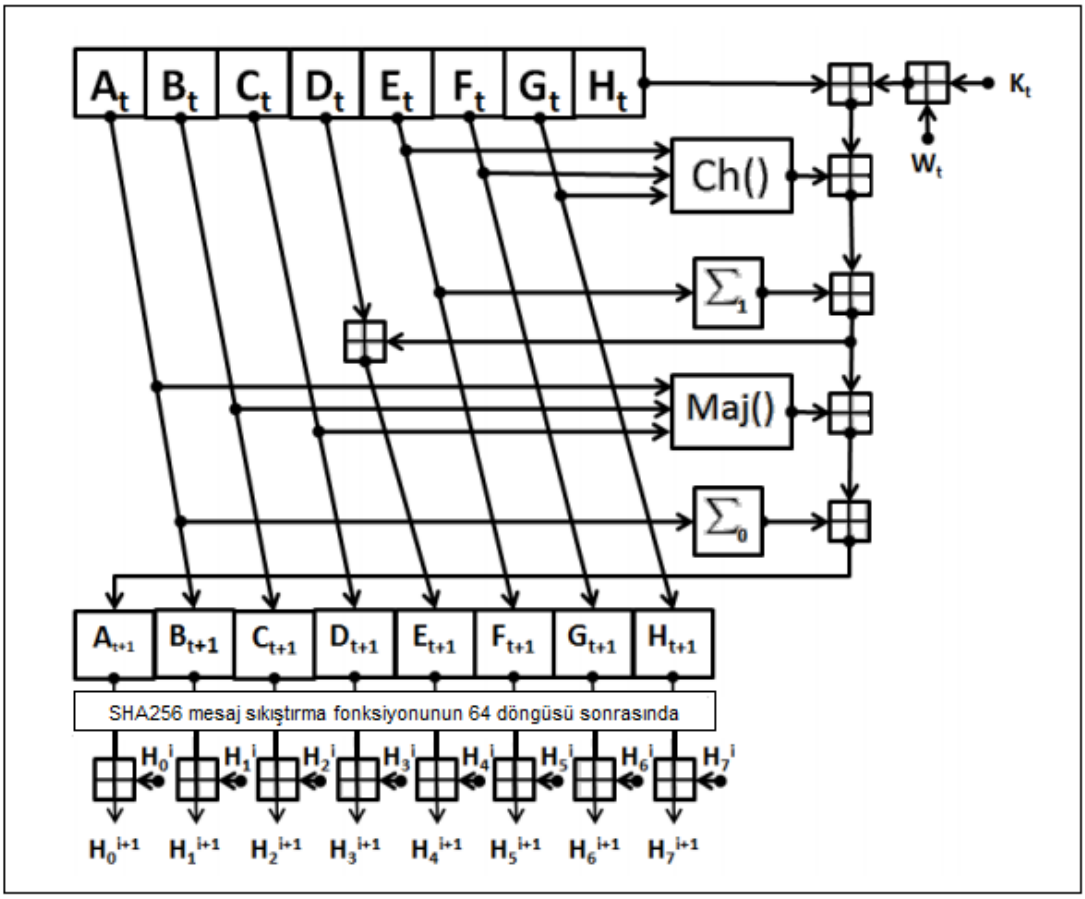
$$\Sigma_0(X) = ROTR^2(X) \oplus ROTR^{13}(X) \oplus ROTR^{22}(X) \quad (3.14)$$

$$\Sigma_1(X) = ROTR^6(X) \oplus ROTR^{11}(X) \oplus ROTR^{25}(X) \quad (3.15)$$

Eşitlik (3.12-3.15)'de yer alan mantıksal işlevlerden Ch ve Maj, girdi olarak 3 kelime almaktadır ve tek bir kelime çıktısı üretmektedir. ^ simgesi, 32 bitlik 'VE' işlemini, ¬ simgesi, işleme giren değerin tümleyenini bulmaktadır. Maj işlevi her zaman A, B ve C'yi girdi olarak alırken, Ch işlevi daima E, F ve G çalışma değişkenlerini girdi olarak almaktadır. A ve E değişkenleri, her turda hesaplanmaları gereken değişkenlerdir. $\Sigma_0$ ve $\Sigma_1$ işlevleri, her zaman A ve E değişkenlerini girdi olarak alırlar. Böylece mesaj sıkıştırma fonksiyonunda iki kısma bölen bir çeşit simetri görebiliriz.

Şekil 3.4'de sıkıştırma fonksiyonunun farklı bir görünümü temsil edilmektedir, ancak aynı mesaj iletilmektedir. Bu şekilden çıkarılacak nokta, sıkıştırma fonksiyonu 64 kez uygulandıktan sonra yani 64 döngü tamamlandıktan sonra A'dan H'ye kadar

olan çalışma değişkenlerinde bulunan değerler, sıkıştırma fonksiyonuna başlangıçta beslenmiş olan 8 kelimelik veriye eklenmektedir.




Şekil 3.4 : SHA256 mesaj sıkıştırma fonksiyonunun detaylı görünümü.

Başlangıçta beslenmiş bu değerler, SHA256 için sabit başlatma vektörü veya bir ara mesaj özeti olabilmektedir. Bunun nedeni, SHA256 algoritmasının girişin sonunda çıktıya eklendiği Davies-Meyer yapısını izlemesi gerçeğidir. Ara/son özet değerinin bulunduğu eşitlik şu şekildedir.

$$H_0^{i+1} = A + H_0^i;\ H_1^{i+1} = B + H_1^i;\ H_2^{i+1} = C + H_2^i;\ H_3^{i+1} = D + H_3^i$$
$$H_4^{i+1} = E + H_4^i;\ H_5^{i+1} = F + H_5^i;\ H_6^{i+1} = G + H_6^i;\ H_7^{i+1} = H + H_7^i \quad (3.16)$$

N. mesaj bloğunu da içeren tüm mesaj blokları yukarıda anlatıldığı gibi işlem gördükten sonra elde edilen son özet değeri aşağıdaki şekilde temsil edilmektedir.

$$SHA256(M) = H_0^N \,||\, H_1^N \,||\, H_2^N \,||\, H_3^N \,||\, H_4^N \,||\, H_5^N \,||\, H_6^N \,||\, H_7^N \quad (3.17)$$

Özet algoritmasına beslenen mesaj, genişletme dahil 512 bitten küçük veya eşitse, özet işleminin başlatılması için beslenen değer başlatma vektörü olarak

adlandırılmaktadır. Mesajın uzunluğu 512 bitten büyükse özet işlemini başlatmak için önceki 512 bitlik bloğun özet değeri ara girdi olarak kullanılmaktadır. Bir önceki bloğun ara özet değerinin bir sonraki bloğun özet hesaplamasına başlatma vektörü olarak beslendiği bu düzene Merkle-Damgard yapısı adı verilir. SHA256, Merkle-Damgard Paradigması adı verilen bu yapıyı temel alır ve bu yapı çarpışma direnci özelliğine sahip olduğundan dolayı SHA256 özet algoritmasının altında yatan mesaj sıkıştırma fonksiyonu da bu bilgi doğrultusunda bu özelliğe sahiptir.

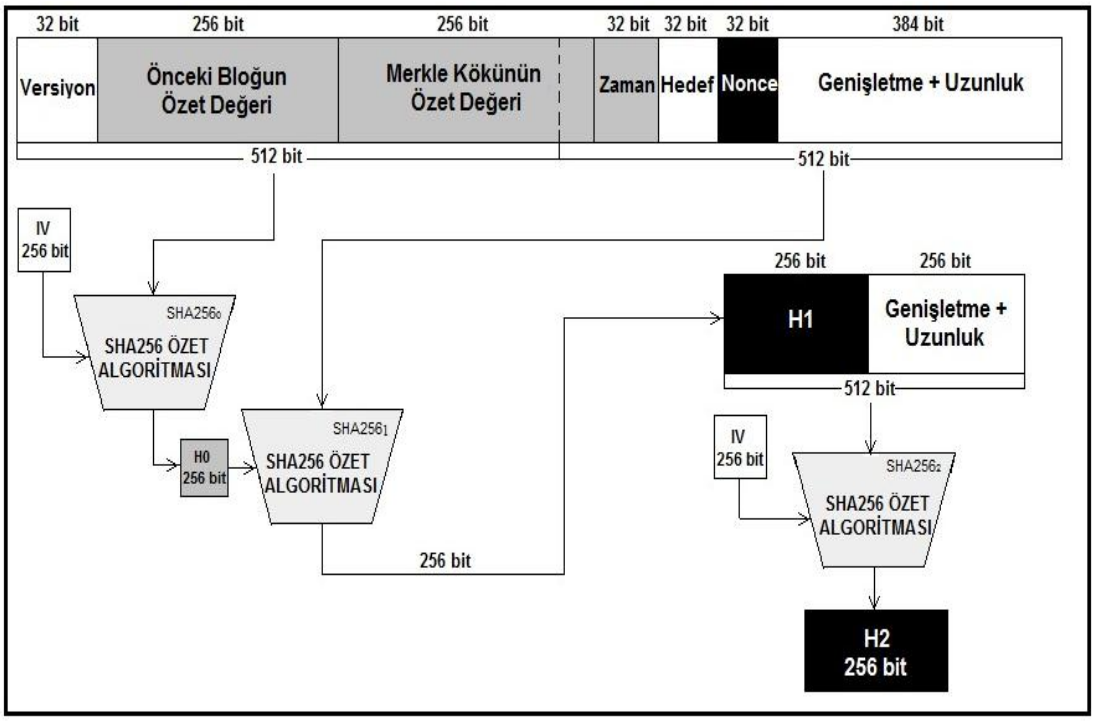
# 4. BİTCOİN MADENCİLİĞİ VE TEMEL İŞLEMLERİ

Bitcoin madenciliği, temel olarak blok başlığının çift SHA256 algoritması ile elde edilen özetinin başlangıç kısmında belli sayıda sıfır olmasını inceleyen süreçtir. Özet değerinin başlangıç kısmındaki sıfırların sayısı hedefin mevcut değerinin zorluk değerine bölünmesi ile elde edilir. Temel olarak yapılan işlem bloğun başlık kısmına çift SHA256 algoritması uygulanarak hedeflenen değeri bulmaktır. Bu işlemin detayları aşağıda ele alınacaktır.

## 4.1 Bitcoin Blok Başlığı Özet Algoritması

Madencilik aygıtları, yeni bir blok bulmak için Bitcoin Blok Başlığı Özet Algoritmasını kullanır ve böylelikle yeni Bitcoinler çıkarılır. Tam olarak teknik bir perspektiften bakıldığında, Bitcoin madencilik aygıtları sürekli bir şekilde blok başlığının çift SHA256 algoritmasını kullanarak özet değerini üretmekte ve Bitcoin ağı tarafından kabul edilmesini beklemektedir. Bu bölümde Bitcoin blok başlığının bileşenlerinin ne olduğu ve nasıl oluşturulduğu konuları üzerinde durulacaktır. Bitcoin blok başlığı yapısı, SHA 256 algoritmasına giren verinin nasıl özetlendiği konusunu açığa kavuşturmaktadır. Esasında tam olarak denilmek istenen özet fonksiyonuna giren verinin madencilik işlemi boyunca ne kadarının sabit kaldığı, çok az değiştiği ve sıklıkla değiştiğinin belirlenmesidir [11].

Bitcoin blok başlığı özet algoritması renk kodlamalı yaklaşım ile anlatılacaktır. Bitcoin madenciliği sürecinde yer alan değerlerin değişiminin belirlenmesinde üç farklı renk olan beyaz, gri ve siyah kullanılmıştır. Beyaz renk süreç boyunca değerin ya hep veya önemli bir süre zarfı içerisinde sabit olduğunu belirtmektedir. Gri renk, değerin değiştiğini fakat çok nadir bir şekilde olduğunu belirtmektedir. Siyah renk ise değerin sürekli olarak değiştiğini (örneğin her özet hesaplamasında) belirtmektedir. Bitcoin ağı tarafından kabul edilen özet değerini elde etmek için girdi verisi olan Bitcoin başlık değerinin yapısı ve SHA256 özet algoritmasına nasıl beslendiği Şekil 4.1'de görülmektedir.




Şekil 4.1 : Bitcoin blok başlığı özet algoritması.

Şekil 4.1'de görüldüğü üzere Bitcoin madenciliğinde blok başlığı üç adet SHA256 özet algoritması işlemine tabi tutulmaktadır. Bu üç özet fonksiyonu sırasıyla $SHA256_0$, $SHA256_1$, $SHA256_2$ olarak tanımlanmıştır. Blok başlığının boyutu 512 bitten büyük olduğu için iki özet fonksiyonuna $SHA256_0$ ve $SHA256_1$ beslenmiştir. $SHA256_0$, ilk 512 bitlik bloğu girdi olarak almıştır ve 64 döngü sonucunda ara mesaj özeti H0 üretilmiştir. $SHA256_0$ 256 bitlik ve önceden tanımlanmış başlatma vektörünü kullanmaktadır. Bu önceden tanımlanmış başlatma vektörü sabit olduğu için beyaz renk ile gösterilmiştir. H0 ara mesaj özetinin hesaplanması tamamen girdilere bağlı ve girdilerinde gri renkli olmasından dolayı H0 değeri de gri renk ile gösterilmiştir.

$SHA256_1$, başlatma vektörü olarak H0 ara mesaj özetini girdi olarak ise diğer kalan 512 bitlik bloğu kullanmaktadır. Girdi bloğunda yer alan nonce değeri siyah ile simgelendiğinden $SHA256_1$ özet fonksiyonunun çıktısı olan H1'de siyah ile gösterilmiştir. Blok başlığının özetinin hesaplanması işlemi burada bitmemektedir. $SHA256_1$ fonksiyonunda üretilen mesaj özeti bir başka özet fonksiyonu olan $SHA256_2$ fonksiyonuna beslenmektedir. $SHA256_2$ özet fonksiyonu, mesaj girdisi olarak 256 bitlik H1 değerini almakta ve 512 bitlik blok haline getirmek için uygun bir genişletme işlemi kullanılmaktadır. Başlatma vektörü olarak daha önce $SHA256_0$ tarafından kullanılmakta olan ve beyaz renk ile simgelenmiş önceden tanımlanmış

başlatma vektörü kullanılmaktadır. $SHA256_2$ özet fonksiyonunun 64 döngüsü sonucunda son özet değeri olan H2 üretilir ve girdilere bağlı olarak siyah renk ile simgelenir. Sonrasında H2 değerinin Bitcoin protokolünün mevcut kısıtlamalarını karşılayıp karşılamadığına bakılır. Eğer H2 değeri mevcut kısıtlamaları sağlıyorsa, doğru nonce değeri ile birlikte elde edilen blok kabul işlemi için hemen Bitcoin ağında yayımlanır. Bunun sonucunda Bitcoin madenciliğinin temel olarak aşağıda yer alan hesaplamayı değişken olan nonce değeri ile birlikte milyarlarca defa yapılması olarak tanımlayabiliriz [11].

$$H2 = SHA256(SHA256(\text{blok başlığı})) \quad (4.1)$$

Burada akıllara şu soru gelebilir; sonda yapılan SHA256 işlemi niçin ek olarak yapılmıştır. Bir anlatıma göre Satoshi Nakamoto'nun çift SHA256'yı tercih etmesinin nedeni uzunluk genişletme saldırısını önlemek olduğu ileri sürülmüştür [11]. SHA256 özet algoritması, Merkle-Damgard yaklaşımını kullanmaktadır ve bu saldırıya karşı açıktır. Bu saldırı şu şekildedir; eğer saldırgan tarafından SHA256(x) biliniyorsa, x girdisi hakkında bir bilgiye sahip olmadan SHA256(x||y)'yi hesaplayabilir. Bitcoin protokolünün bu saldırının zararına karşı ne kadar duyarlı olduğu tamamen bilinmemesine rağmen Satoshi Nakamoto'nun güvenliği sağlamak için çift SHA256 kullandığı düşünülmektedir. Başka bir söyleme göre çift özet fonksiyonu ile 128 döngülük SHA256 elde edilmekte ve gelecekte SHA256'ya karşılık geliştirilebilecek saldırılara karşılık sistemin güvenilirliğinin sağlandığı ileri sürülmektedir.

### 4.2 Bitcoin Blok Başlığının Detayları

Blok başlığı, madencilik işlemi sırasında arada bir güncelleme ihtiyacı duymaktadır. Mevcut olan işlemler blok başlığında değil, bloğun ana yapısında yer almaktadır. Blok başlığında ise dolaylı olarak işlemlerin bir özet değeri olan Merkle kökü yer almaktadır. Bu akılcı metot ile işlemlerin bütünlüğünün sağlanması yanında blok başlığının blokta yer alan işlemlerin sayısından bağımsız olması temin edilmektedir [12]. Blok başlığının içerdiği alanlar şu şekilde tanımlanabilir;

**Versiyon:** 32 bitlik bu değer, Bitcoin yazılımının yeni bir bloğu üretmek için takip ettiği kuralların versiyonunu temsil etmektedir. Bu değerin sabit olduğu bilindiği için Çizelge 4.1'de beyaz renk ile kodlanmıştır.

**Önceki bloğun özet değeri:** Bitcoin ağı tarafından kabul edilmiş bloğun 256 bitlik özet değeridir. Yeni blok başlığında bu değer yer alarak madenci tarafından en uzun blok zincirinin genişletilmesi amaçlanmaktadır. En son kabul edilen bloktan sonra madenci iş kanıtı problemini ilk olarak çözmek için denemelere başlar. İş kanıtı probleminin çözümünde çeşitli madenciler arasında ciddi bir yarış mevcuttur ve her birisi problemi çözen ilk kişi olmayı ve Bitcoin ağına çözümü yayımlamayı amaçlamaktadır. Eğer yayımlamış olduğu çözüm Bitcoin ağı tarafından kabul edilirse buna karşılık 12.5 BTC ödül kazanmaktadır. Bitcoin sistemi temel olarak yaklaşık her 10 dakikada bir ağ tarafından blok üretilmesi prensibine göre tasarlanmıştır. Bundan dolayı yaklaşık olarak bir önceki bloğun özet değeri 10 dakikada bir güncellenmelidir ve bu nedenle bu alan gri olarak kodlanmıştır.

Çizelge 4.1 : Bitcoin blok başlığının görünümü.

| Alan | Boyut | Tanım |
|---|---|---|
| Versiyon | 32 bit | Bitcoin versiyon bilgisi Bitcoin yazılımına dayanmaktadır. |
| Önceki bloğun özet değeri | 256 bit | Bitcoin ağı tarafından kabul edilmiş bir önceki bloğun özet değeridir. |
| Merkle kökünün özet değeri | 256 bit | Bitcoin işlemlerinin özet değerlerinin bulunduğu Merkle ağacındaki kök değeridir. |
| Zaman Damgası | 32 bit | Mevcut zaman damgası, 1970-01-01 T00:00 UTC'den bu yana ki saniye miktarıdır. |
| Hedef | 32 bit | Mevcut hedef değeri 32 bitlik bir sayı ile gösterilir. |
| Nonce | 32 bit | Her bir özet fonksiyonu denemesinden sonra artırılır, 0x00000000-0xFFFFFFFF değeri arasında değişen bir sayıdır. |
| Genişletme + Uzunluk | 384 bit | Standart SHA256 genişletme değeri veri üzerine eklenir. |

**Merkle ağacı:** Bitcoin blok zincirindeki her blok, bir Merkle ağacını kullanarak bloğun tüm işlemlerinin bir özetini içermektedir. İkili özet ağacı olarak da bilinen bir Merkle ağacı, büyük veri kümelerinin bütünlüğünü verimli bir şekilde özetlemek ve doğrulamak için kullanılan bir veri yapısıdır. Merkle ağaçları kriptografik özet fonksiyonlarını içeren ikili ağaçlardır. Ağaç terimi bilgisayar bilimlerinde dallanmış bir veri yapısını tanımlamak için kullanılmaktadır, ancak bu ağaçlar genelde aşağıdaki örneklerde görüleceği gibi baş kısmında kök ve şemanın alt kısmında yapraklardan oluşmaktadır.

Merkle ağaçları, Bitcoin sisteminde yer alan herhangi bir bloğun tüm işlemlerini özetlemek için kullanılmaktadır ve tüm işlem setinin genel bir parmak izi üretilmektedir ve bir işlemin bir bloğa dahil edilip edilmediğini doğrulamak için çok verimli bir süreç sağlamaktadır. Bir merkle ağacı, düğüm çiftlerinin yinelemeli olarak özetinin alınması ile oluşmaktadır ve bu işlem sonucunda en son tek bir özet değeri kalmaktadır ve bu değere de Merkle kökü adı verilmektedir. Bitcoin sisteminde Merkle ağaçları için kriptografik özet fonksiyonu olarak SHA256 kullanılmaktadır ve iki kez uygulanmasından dolayı çift SHA256 olarak adlandırılmaktadır. N adet verinin özeti alınarak bir Merkle ağacı içerisinde depolandığında, ağacın içerisine herhangi bir veri öğesinin eklenip eklenmediğinin kontrolü en fazla $2*\log_2(N)$ adımda gerçekleştirilebilir, bundan dolayı Merkle ağacı oldukça verimli bir veri yapısı olarak göze çarpmaktadır.

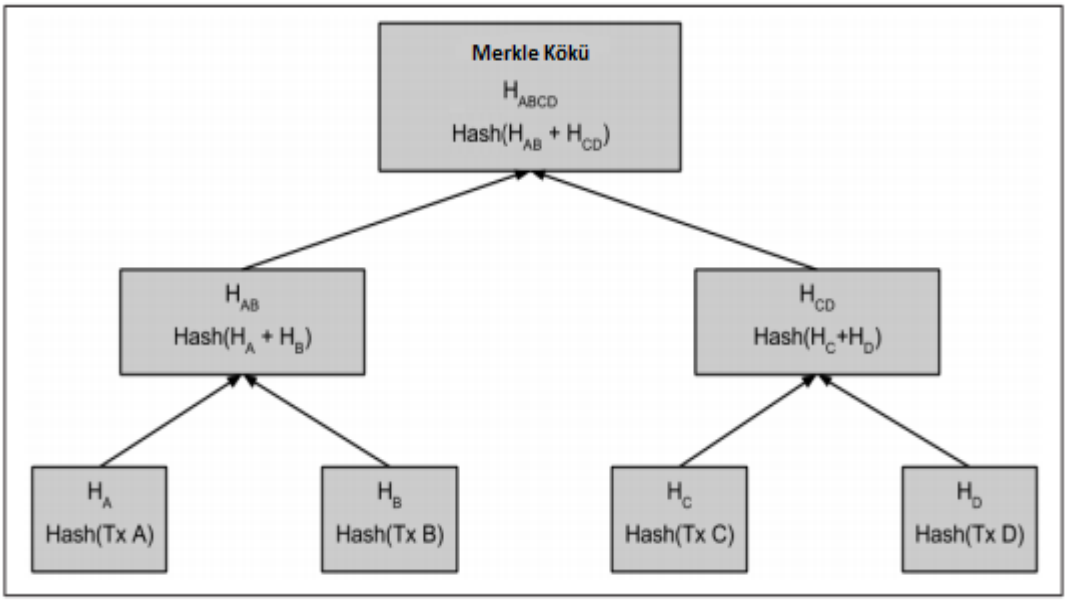
Merkle ağacı, aşağıdan yukarıya doğru inşa edilmiştir. Aşağıdaki örnekte, şeklin alt kısmında gözüken ve Merkle ağacının yapraklarını oluşturan dört işlem A, B, C ve D ile başlıyoruz. İşlemler merkle ağacında depolanmaz, daha ziyade verinin özeti alınarak ortaya çıkan özet değeri her bir yaprak düğümünde $H_A$, $H_B$, $H_C$ ve $H_D$ olarak depolanır.

$$H_A = SHA256(SHA256(Transaction\ A)) \quad (4.2)$$

Yaprak düğümlerin ardışık çiftlerinin sahip oldukları özet değerleri birleştirilerek tekrar yinelemeli olarak özet değeri alınır ve bu değer bir ana düğümde depolanır. Örneğin, üst düğüm $H_{AB}$'yi oluşturmak için, çocuklarının iki 32 baytlık özeti, 64 baytlık bir veri oluşturmak üzere birleştirilir. Bu veri üst düğümün özet değerini üretmek üzere çift SHA256 fonksiyonuna beslenmektedir.

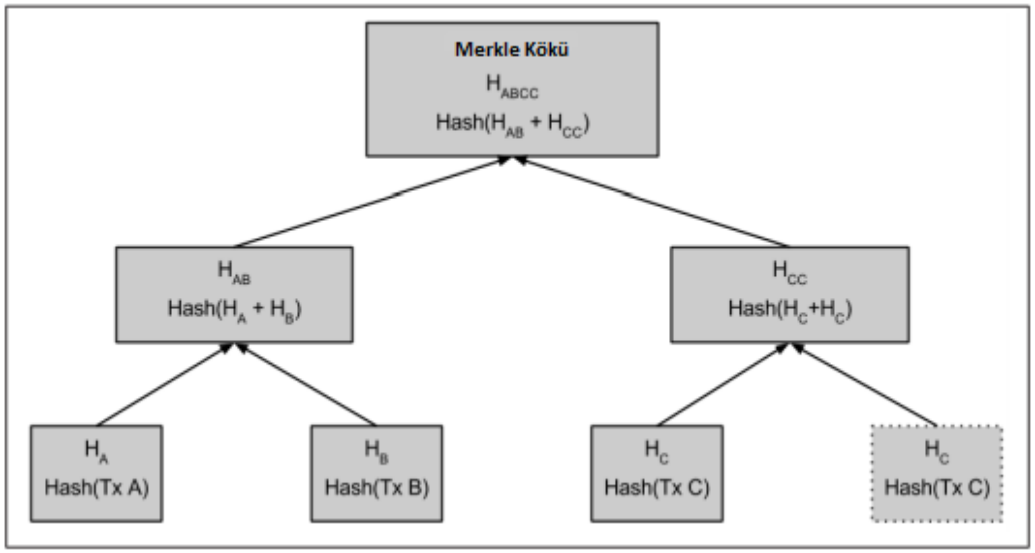
$$H_{AB} = SHA256(SHA256(H_A + H_B)) \quad (4.3)$$

İşlem, merkle ağacının en üstünde yalnızca bir düğüm bulunana kadar devam etmektedir ve bu düğüm merkle kökü olarak bilinmektedir. Bu tür bir veri yapısının görünümü Şekil 4.2'de verilmiştir. Bu 32 baytlık özet, blok başlığında saklanır ve dört işlemin tüm verisini özetlemektedir.




Şekil 4.2 : Bir Merkle ağacındaki düğümlerin hesaplanması.

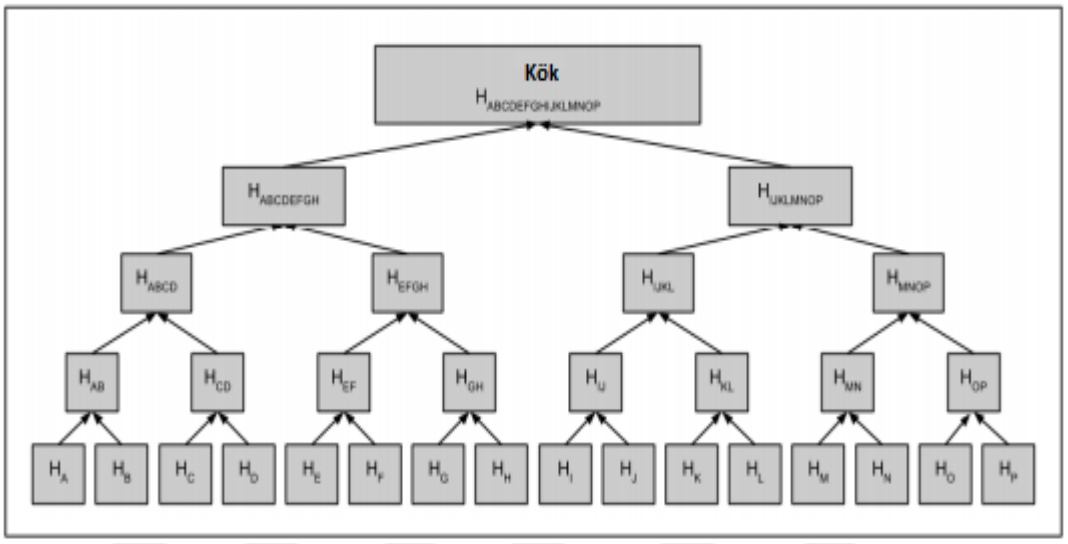
Merkle ağacı ikili bir ağaç olduğundan, çift sayıda yaprak düğümü gerektirir. Eğer özetlenecek işlem sayısı tek sayıda ise, son işlemin özet değeri tekrarlanarak çift sayıda yaprak düğümü oluşturulmaktadır ve bu tür yapı dengeli ağaç olarak adlandırılmaktadır. Bu tür bir durumun olduğu merkle ağacı Şekil 4.3'de gösterilmiştir ve çift sayıda yaprak düğümü olması için C işleminin özet değeri tekrarlanmıştır.




Şekil 4.3 : Çift sayıda yaprak düğümünün olması için bir veri elemanının çoğaltılması.

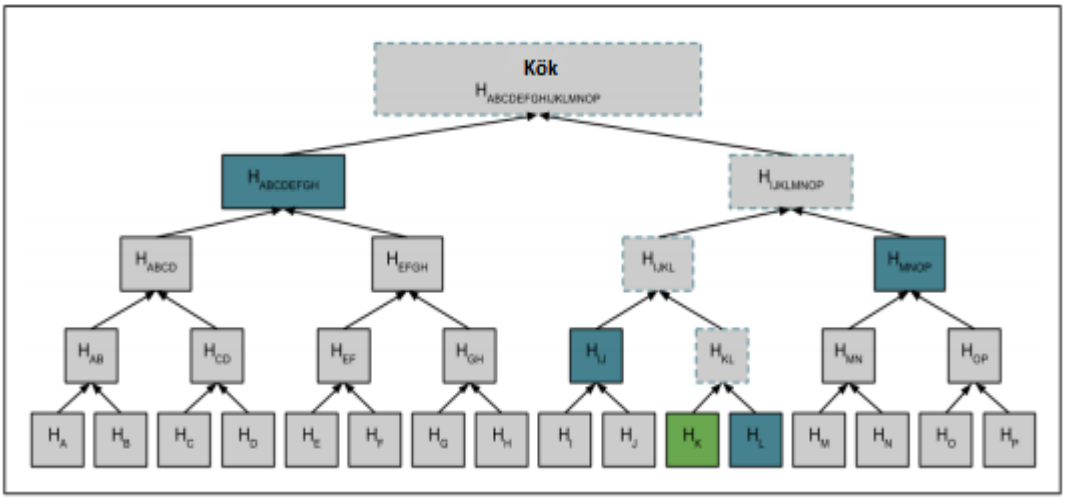
Dört işlemden bir ağacın oluşturulması için kullanılan yöntem, herhangi bir boyuttaki ağaçları oluşturmak için genelleştirilebilir. Bitcoin'de yalnız bir blok içerisinde

binden fazla işlem bulunmaktadır ve bu işlemlerin özet değeri alınarak benzer şekilde sadece 32 baytlık bir değere sahip olan merkle kökü elde edilmektedir. Şekil 4.4'de 16 işlemden oluşan bir ağaç yapısı görülmektedir. Merkle kökü şekilde yaprak düğümlerinden daha büyük gözükse de veri boyutu olarak 32 bayttan oluşmaktadırlar. Blokta bir işlem veya binlerce işlem olmasına bağlı olmadan merkle kökü her zaman 32 baytlık bir değere sahiptir.




Şekil 4.4 : Çok sayıda veri elemanının bulunduğu Merkle ağacının görünümü.

Belirli bir işlemin blokta var olup olmadığını anlamak için, sadece $\log_2(N)$ adet 32 baytlık özet değerine ihtiyaç duyulmaktadır. Bu özet değerlerinin oluşturduğu düğümler kimlik doğrulama ya da merkle yolu olarak adlandırılmaktadır ve bu düğümler ilgili işlemin köke ulaşmasını sağlamaktadır. Esasında işlem sayısının arttığı durumda oldukça önemlidir, çünkü 2 tabanında logaritma değeri işlem sayısının artmasına göre oldukça yavaş değişim göstermektedir. Bu da bitcoin düğümlerinin en fazla on veya on iki özet değerinden oluşan kimlik doğrulama yoluna sahip olmasına neden olmaktadır ve böylelikle bir işlemin blokta var olup olmadığına megabayt boyutundaki blokların incelenmesine gerek kalmadan ulaşılabilmektedir. Şekil 4.5'de görüldüğü gibi, K işleminin blokta olup olmadığını saptamak için 4 adet 32 bayt özet değerine ihtiyaç duyulmaktadır. Bu yol dört adet özet değeri içermektedir ve bu değerler $H_L$, $H_{IJ}$, $H_{MNOP}$ ve $H_{ABCDEFGH}$'dir. Kimlik doğrulama yolunda bu dört özet değeri, çiftleri ile birleştirilerek özet değerleri alınır ve eğer merkle köküne ulaşılırsa ilgili işlemin blokta olduğu sonucuna ulaşılır.




Şekil 4.5 : Bir veri öğesinin dahil edildiğini kanıtlamak için kullanılan bir Merkle yolu.

Merkle ağaçlarının etkinliği, ölçek arttıkça belirgin hale gelmektedir. Örneğin, bir işlemin bir bloğun parçası olduğunu kanıtlamak için oluşan veri boyutu Çizelge 4.2'de özetlenmiştir.

Çizelge 4.2 : Merkle Ağaçlarının Verimliliği.

| **İşlem Sayısı** | **Yaklaşık olarak bloğun boyutu** | **Yol Boyutu (Özet)** | **Yol Boyutu (Bayt)** |
|---|---|---|---|
| 16 | 4 KB | 4 | 128 |
| 512 | 128 KB | 9 | 288 |
| 2048 | 512 KB | 11 | 352 |
| 65536 | 16 MB | 16 | 512 |

Çizelge 4.2'den görülebileceği gibi, blok büyüklüğü oldukça hızlı bir artış göstermekte iken bir işlemin dahil edildiğini kanıtlamak için gereken merkle yolu çok daha yavaş artmaktadır. Blok zincirinin gigabayt seviyesindeki verilerini depolamadan veya iletmeden, Merkle ağaçları yardımı ile bir işlemin blokta olup olmadığına küçük boyutlardaki verilerden ulaşılabilmektedir.

**Merkle kökünün özet değeri**: Blok içerisinde yer alan işlemlerin özet değerinin tutulduğu Merkle ağacının kökü olan 256 bitlik değerdir. Yeni bir blok bulunduğunda veya 32 bitlik nonce değerinin hepsi için özet işlemi yapılmasına

rağmen hedef değeri sağlanamamışsa değişmektedir ve bu nedenle gri renk ile kodlanmıştır [5].

**Hedef:** İlk kısımda anlatılan hedef değeri ile aynıdır, sadece o değerin 256 bitten 32 bite sıkıştırılmış halidir [11]. Hedef değeri her 2016 yeni bloktan sonra değişmektedir ve bu da yaklaşık iki haftaya tekabül etmektedir. Bundan dolayı bu alanda beyaz renk ile kodlanmıştır.

Sistem tarafından belirtilen 32 bitlik hedef değerinden karşılaştırma için kullanılacak 256 bit şu şekilde elde edilir. Örneğin, Bitcoin sistemi tarafından 32 bitlik hedef değeri 0x1903a30c olarak verilsin. Bu değerin ilk iki onaltılık basamağı üs, diğer kalan altı basamak ise katsayı olark adlandırılmaktadır. Yani, bu bloğun doğrulanması için verilmiş hedef değerinin üs kısmı 0x19, katsayı kısmı ise 0x03a30c olarak bulunmaktadır. Hedef değerinin 256 bitlik gösterimi aşağıdaki eşitliğe göre bulunmaktadır [1].

$$\text{Hedef} = \text{Katsayı} * 2^{\wedge} ( 8 * ( \text{Üs} - 3 ) ) \quad (4.4)$$

Bu eşitlik kullanılarak 0x1903a30c değerinin 256 bitlik gösterimi aşağıdaki işlemler sonucunda onaltılık tabanda elde edilmektedir.

$$\text{Hedef} = 0x03a30c * 2^{(0x08*(0x19-0x03))}$$

Böylelikle, iş kanıtı problemlerinde karşılaştırılacak olan hedef değeri bulunmuştur ve blok başlığının iki SHA256 özet algoritmasından elde edilen sonuç bu değer ile karşılaştırılarak yeni blokların doğrulanması işlemi gerçekleştirilecektir.

**Nonce:** Bu 32 bitlik değer, blok başlığının çift SHA256 özet fonksiyonu ile özeti alındığında değişmektedir. Çok sık bir değişim söz konusu olduğu için siyah renk ile kodlanmıştır. Bu değer 0'dan başlar ve lineer olarak doğru nonce değeri bulununcaya kadar arttırılır.

Bitcoin madenciliğinde iş kanıtı probleminin çözümü bulununcaya kadar nonce değeri değişmektedir ve sistem tarafından belirtilen hedef değerine ulaşıncaya kadar her bir denemede bir birimlik artış yapılmaktadır. Nonce değeri 0 ile $2^{32}$ arasında bir sayı olduğundan iş kanıtı problemi $2^{32}$ denemede eğer bulunmazsa nasıl bir yol izlenildiğini şu şekilde açıklayabiliriz.

Bitcoin sisteminin başladığı ilk zamanlarda maden işlemi yapan cihazların özet üretme hızı oldukça düşük miktardadır. Bundan dolayı zaman damgasında oluşabilecek bir değişim olmadan bütün nonce değerlerinin denenmesi ile iş kanıtı problemi için çözüm bulunamaz. Bu nedenden ötürü sistemin başladığı ilk zamanlarda nonce değerinin hepsi denenmesine rağmen problemin çözümü bulunamadığında zaman damgası değiştirilerek yeni blok başlığı değeri elde edilir ve bu değer ile problem çözümü gerçekleştirilir.

Günümüzde ise maden işlemi yapan cihazlar oldukça yüksek özet üretme hızına sahiptirler. Bundan dolayı zaman damgasında bir değişim meydana gelmeden nonce değerinin sahip olduğu tüm aralık problem çözümü için denenmiş olmaktadır. Bu sorunun çözümü ise şu şekildedir: Her bir madenci seçmiş olduğu işlemlerden bir blok oluştururken ilk işlem olarak kendi adres bilgisini içeren ve bazı sabit alanlar bulunan değeri bloğa ekleyerek iş kanıtı problemini çözdüğünde bu değerde yer alan adres bilgisine göre sistem tarafından kendisine ödeme yapılmaktadır. Bu ilk işlem Bitcoin sisteminde üretim işlemi olarak adlandırılmaktadır. Üretim işleminde yer alan bileşenler Çizelge 4.3'de verilmiştir [1].

Çizelge 4.3 : Üretim işleminde yer alan bileşenler.

| **Boyut** | **Bileşen** | **Açıklama** |
|---|---|---|
| 32 bayt | İşlem özet değeri | Tüm bitlerin değeri 0'dır. |
| 4 bayt | Çıktı indeksi | Tüm bitlerin değeri 1'dir. |
| 1-9 bayt | Madenci tarafından yeni bir blok doğrulandığında alınacak ödül ve işlem ücretlerinin yer aldığı bilginin boyutu | 2 ile 100 bayt arasında değişen bilgi kısmıdır. |
| Değişken | Madenci tarafından yeni bir blok doğrulandığında alınacak ödül ve işlem ücretlerinin yer aldığı bilgi | Keyfi bir değerdir ve madencilik için gerekli olan ekstra nonce değeri bu keyfi değerin içinde yer almaktadır. |
| 4 bayt | Sıra numarası | Tüm bitlerin değeri 1'dir. |

Çizelge 4.3'de görüldüğü üzere 2 ile 100 bayt arasında herhangi bir değer içeren bir kısım bulunmaktadır ve bu kısım içerisinde ekstra nonce adı verilen bir bileşen oluşturulmuştur. Eğer madenci yeni bir bloğu $2^{32}$ denemede doğrulayamamışsa, bu kısımda yer alan ekstra nonce değeri bir birim artırılarak merkle kökünün yeni bir değer almasını böylelikle de yeni bir blok başlığını elde etmektedir. Yeni blok başlığı değeri kullanılarak iş kanıtı probleminin çözümüne devam edilmektedir.

**Genişletme + Uzunluk:** $SHA256_1$ için, genişletme + uzunluk toplam 384 bit iken $SHA256_2$ için 256 bittir. $SHA256_1$ ve $SHA256_2$ fonksiyonlarının girdi mesaj uzunluklarının 640 ve 256 bit olduğu bilindiği için genişletme + uzunluk değerleri de sabittir. Bundan dolayı bu alan beyaz renk ile kodlanmıştır [11].

Tüm bu anlatılanlardan sonra akla her bir madencinin aynı özet değerini üreteceği düşüncesi gelebilir. Ancak bu tam olarak doğru değildir. Çünkü her bir madencinin seçmiş olduğu işleme göre farklı bir Merkle kökü elde edilmektedir ve böylelikle her bir madencinin bulmuş olduğu özet değeri farklıdır. Bundan dolayı çözüm eşsiz olduğu için ağda yer alan her bir madencinin çözümü bulma şansı eşittir. Ancak madenciler, daha yüksek verimlilikte özet bulma işlemi yapan aygıtlara sahip olurlarsa problemi çözme şanslarını arttırabilirler. Fakat bu oldukça yüksek maliyetler gerektirdiği için günümüzde kişiler madencilik havuzlarından hisse alarak Bitcoin sisteminden para kazanmayı tercih etmişlerdir.

# 5. ÖZET FONKSİYONU İÇİN LİTERATÜRDE ÖNERİLMİŞ OPTİMİZASYONLAR

Literatür araştırması sonucu incelenen çalışmalardan [11]'de özet fonksiyonu için optimizasyon yöntemleri önerilmiş ve bu yöntemlerin hepsi özet fonksiyonun girdisi olan Bitcoin blok başlığının bazı alanlarının sabit olması veya sıfır olmasına bağlı olarak ortaya çıkarıldığı görülmüştür. İleri sürülen bu optimizasyonlar detaylı bir şekilde açıklanmıştır.

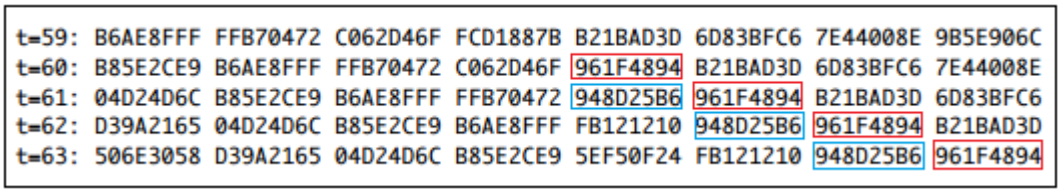
## 5.1 $SHA256_0$ için H0 Hesaplanması

Şekil 4.1'e bakıldığında $SHA256_0$ fonksiyonuna giren değerler; versiyon, önceki bloğun özet değeri ve Merkle kökünün özet değeri alanları olduğu görülmektedir [11]. Bu alanlardan 32 bitlik versiyon değeri sabit olduğu için beyaz renklidir, benzer şekilde 256 bitlik önceki bloğun özet değeri ve 224 bitlik Merkle kökü de yeni bir blok bulunmadığı sürece sabittir ve gri renkle gösterilmiştir. Bundan dolayı $SHA256_0$ işleminin yeni bir blok bulunmadığı sürece tekrar yapılmasına gerek yoktur. Bu tür bir optimizasyon günümüzdeki madencilik aygıtlarında kullanılmaktadır. Bu yüzden sonuç değeri bulunurken bu optimizasyon hesaba katılmayacaktır.

## 5.2 $SHA256_2$'de önceden özet değerinin kontrolünün yapılması

$SHA256_2$ işlemi sonucunda elde edilen özet değerinin hedef değerinden küçük olup olmadığı kontrol edilir. Özet değerinin hedef değerinden küçük olmasını belirleyen değişken g ve belirli sayıda 0 içermesini gösteren değişken de h'dir. Algoritmaya göre bu işlem tüm döngüler sona erdiğinde yapılmaktadır. Ancak elde edilen özet değerinin beklenen sayıda 0 içermesi ve hedef değerinden küçük olmasının kontrolü 61. ve 62. döngüdeki e değişkeninin değerine bakılarak bulunabilir. Bu sonuca SHA256 algoritmasındaki Eşitlik (3.7-3.8) yardımı ile ulaşılmıştır. İşlemlere bakıldığında özet değerinin son değerleri olan f, g ve h değişkenlerinin e'ye bağlı olduğu görülmektedir ve e değişkeni her dört döngüde bir sırasıyla f, g ve h değişkenlerine atanmaktadır. Bundan dolayı 64. döngüdeki h değişkenine 61.

döngüde, g değişkenine de 62. döngüde ulaşılmaktadır. Bu sonucu destekleyen Şekil 5.1'de aşağıda görülmektedir, Ulusal Standartlar ve Teknoloji Enstitüsü tarafından paylaşılmış olan SHA256 örneğinde elde edilen sonuç görülmektedir [11].




Şekil 5.1 : SHA256 algoritmasında değişkenlerin görünümü [13].

Bu kontrol mekanizması ile hesaplanan özet değerinin geçerli olup olmadığına genellikle 61. döngüde karar verilir, böylelikle 3 döngü için işlem yapılmasına gerek kalmamıştır. Çok az durumda ise 62. döngü de yapılmaktadır. Genel olarak 3 döngülük bir iş yükü iyileştirilmesi incelenen yöntemle sağlanmıştır.

### 5.3 $SHA256_1$'in İlk Üç Döngüsü

Şekil 4.1'e bakıldığında $SHA256_1$ işleminin genişletilmiş mesaj bloğunun ilk 3 değerine sırası ile Merkle kökü değerinin son 32 biti, zaman damgası ve hedef değeri beslenmektedir. Bu değerlerden ikisi Merkle kökü ve zaman damgası nadiren değiştiğinden gri renkte ve çoğu zaman değişmediği görülen hedef değeri de beyaz olarak kodlanmıştır. Merkle kökü yeni bir blok eklendiğinde veya tüm nonce değerleri için özet işlemi yapılmasına rağmen hedef değerine ulaşılamadığında değişmektedir, zaman damgası da Merkle kökü değiştiğinde güncellenmektedir. Bundan dolayı yeni bir blok gelmediği sürece bu iki değer aynı kalmaktadır. Hedef değeri de her 2016 yeni blok üretildikten sonra değişmektedir. Bundan dolayı $SHA256_1$'in ilk üç döngüsü bir defa hesaplandıktan sonra her bir nonce değeri için tekrar hesaplanmasına gerek olmadığı ortaya konmuştur [11].

### 5.4 $SHA256_1$'in 4. döngüsünün Artan Olarak Hesaplanması

Şekil 4.1'e bakıldığında $SHA256_1$ işleminin genişletilmiş mesaj bloğunun 4. değişkenine nonce değeri beslenmektedir. 4. döngüde, Eşitlik (3.6-3.11) arasında yapılan işlemlerde yukarıda yapılmış olan optimizasyon sonucu tüm değerler sabittir ancak $W_3$ değeri değişkendir. $W_3$ değişkeninde nonce değeri bulunmaktadır ve her zaman 1 artarak değişir. Bundan dolayı başlangıç nonce değerinde 4. döngü bir defa

hesaplandıktan sonra nonce değerinin her bir artışında 4. döngüde a-h arasındaki tüm değişkenlerin değeri 1 arttırılır. Aşağıda bu durumun görüntüsü yer almaktadır.

| Nonce | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 0x00000000 | c14c28c6 | fdd86aa7 | 1184d36 | 2703413e | 346785c7 | c1abdbc7 | 8f925db9 | a4b56f21 |
| 0x00000001 | c14c28c7 | fdd86aa7 | 1184d36 | 2703413e | 346785c8 | c1abdbc7 | 8f925db9 | a4b56f21 |
| 0x00000002 | c14c28c8 | fdd86aa7 | 1184d36 | 2703413e | 346785c9 | c1abdbc7 | 8f925db9 | a4b56f21 |
| 0x00000003 | c14c28c9 | fdd86aa7 | 1184d36 | 2703413e | 346785ca | c1abdbc7 | 8f925db9 | a4b56f21 |
| 0x00000004 | c14c28ca | fdd86aa7 | 1184d36 | 2703413e | 346785cb | c1abdbc7 | 8f925db9 | a4b56f21 |
| 0x00000005 | c14c28cb | fdd86aa7 | 1184d36 | 2703413e | 346785cc | c1abdbc7 | 8f925db9 | a4b56f21 |

Şekil 5.2 : $SHA256_1$'in 4. döngüsünde yapılan optimizasyon [13].

### 5.5 $SHA256_1$ ve $SHA256_2$'nin genişletilmiş mesaj bloklarından bazılarının 0 olması ve böylelikle toplama işlemlerinin azaltılması

Şekil 4.1'e bakıldığında $SHA256_1$ ve $SHA256_2$ fonksiyonlarına giren mesaj uzunluğunun hep sabit olduğu görülmektedir. Bundan dolayı genişletme + uzunluk kısmında uzunluk değerinin $SHA256_1$ için 640, $SHA256_2$ için 256 olduğu bilinmektedir ve genişletme için gerekli olan toplam 0 sayısı da L(mesaj uzunluğu) + k(0 sayısı) + 1(1bit)'in mod 512'de 448'e eşit olması gerektiğinden sırasıyla 319 ve 191 bit olarak elde edilmektedir. Eşitlik (3.6)'da $W_j$ toplamı yapılırken $SHA256_1$'de $W_5$-$W_{14}$ arasındaki değişkenler, $SHA256_2$'de $W_9$-$W_{14}$ arasındaki değişkenler 0'dır ve $SHA256_1$ fonksiyonunda 10 adet toplama işlemi, $SHA256_2$ fonksiyonunda 6 adet toplama işlemi yapılmayabilir [11].

### 5.6 $SHA256_1$ ve $SHA256_2$ işlemlerinde mesaj uzunluğun sabit olması

Yukarıdaki kısımda da belirtildiği gibi özet fonksiyonlarına giren mesaj uzunluğu sabittir. Bundan dolayı 64 bit ile simgelenen uzunluk değerinin ilk 32 biti sıfırdır ve bu 32 bitin yukarıda bahsedilen optimizasyon işlemi ile yükü ortadan kaldırılmıştır. Geriye kalan 32 bit değeri ise her iki özet işlemi içinde genişletilmiş mesaj bloğunun $W_{15}$ değişkeninde yer almaktadır. Bu değişken sabit olduğu için önceden $K_{15}$ + $W_{15}$ hesaplanarak fonksiyonlara beslenebilir. Böylelikle iki toplama işleminden kurtulmuş olunur. Benzer şekilde 1 bitlik 1 değeri mesajların sonuna eklenmektedir ve genişletme işleminden dolayı 32 bitlik değer 0x80000000 olarak ortaya çıkmaktadır. Bu değer $SHA256_1$'de genişletilmiş mesaj bloğunun $W_4$ değişkeninde yer alırken, $SHA256_2$'de $W_8$'de yer almaktadır. Yukarıdakine benzer şekilde bu değerlerin toplamı da başlangıçta yapılabilir ve fonksiyonlara toplamları beslenebilir.

Bu optimizasyon metodu ile toplamda 4 toplama işleminin ortadan kaldırılması sağlanmıştır.

Yukarıda literatür taraması sonucu elde edilen yöntemlerin donanımda ne kadar bir performans artışı sağlayabileceğini göstermek için FPGA'de Verilog programlama dili yardımı ile Bitcoin madenciliğinde kullanılan algoritma uygulanmıştır. Bir sonraki bölümde FPGA'in detaylarından ve uygulaması gerçekleştirilen Bitcoin madenciliğinin simülasyon ve FPGA sonuçlarının detayları incelenecektir. Simülasyonlar ve FPGA sonuçlarına göre her bir metodun yaptığı katkı detaylı bir şekilde ele alınacaktır. Önerilen bu yöntemlerin FPGA'de gerçekleştirilmesi sonucunda Bitcoin madenciliğinde ciddi bir performans artışı sağlamayı amaçlamaktayız. Şu an için Bitcoin madenciliğinin ASIC aygıtlarında yapıldığı düşünülebilir, ancak FPGA kodunda gerekli optimizasyonlar yapıldıktan sonra bu uygulamanın ASIC'de üretimi gerçekleştirilebilir. Bundan dolayı Bitcoin ağının özet üretimi hızında belli bir seviyede artış meydana gelmesi beklenmekte ve aynı enerji tüketiminde daha çok özet üretilmesi hedeflenmektedir. Yukarıda bahsedilen her bir iyileştirmenin, şu anda mevcut olan uygulamada ne kadar işlem yükünü azalttığı ve performans artışının hangi oranda değiştiği bir sonraki başlık altında incelenecektir.

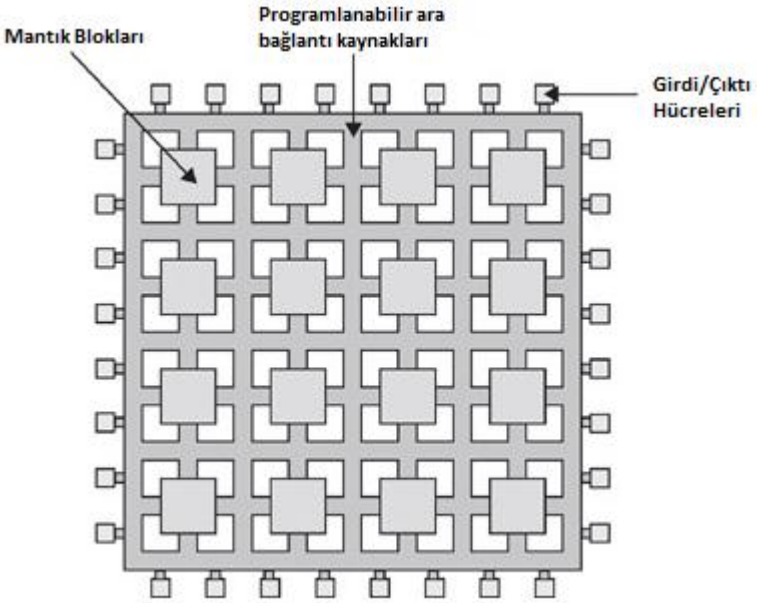
# 6. BİTCOİN MADENCİLİĞİNİN FPGA'DE GERÇEKLENMESİ

Bitcoin madenciliği için başlangıçta CPU'lar kullanılırken sonrasında ise daha hızlı olan ancak çok fazla elektrik tüketen GPU'lar kullanılmıştır. Daha sonrasında ise madenciler FPGA'leri tercih etmişler ancak 2013 yılının ortalarına doğru yalnız bu işlemler için geliştirilmiş olan ve çok yüksek hızlara ulaşabilen ASIC'ler pazardaki yerini almıştır. 5. bölümde literatürde yer alan optimizasyon tekniklerinden bahsedilmiştir ve bu yöntemlerin ASIC'de uygulanabilmesi için öncelikle FPGA (Field Programmable Gate Array)'de gerçeklenmesi ve burada gerçeklenen uygulamada birtakım değişiklikler yapılarak ASIC üretimi yapılabilmektedir. Bu gereksinim doğrultusunda, öncelikle SHA256 özet algoritması, devamında temel fonksiyonu SHA256 olan Bitcoin madencilik sistemi ve literatürde önerilmiş olan optimizasyon teknikleri Verilog programlama dili yardımı ile FPGA'de gerçekleştirilmiştir. Gerçekleştirilen sistem sonucunda yapılan iyileştirmenin performansda nasıl bir değişime neden olduğu incelenecektir. Ayrıca, performans da meydana gelebilecek olumlu bir değişim Bitcoin madenciliğinde oluşan çok büyük miktardaki elektrik tüketimi sorununa önemli ölçüde katkıda bulunacağı düşünülmektedir.

## 6.1 FPGA Hakkında Genel Bilgiler

FPGA'in yapısında yapılandırılabilir mantıksal bloklar ve bu blokları birbirine bağlayan programlanabilir bağlantı elemanları yer almaktadır. FPGA'ler üretici firma tarafından piyasaya sürüldükten sonra programcı veya kullanıcı tarafından herhangi bir uygulama kolayca yapılabilir ve aynı zamanda FPGA'i başka yapmak istediği projelerde de tekrar programlayabilir. Şekil 6.1'de FPGA'in iç mimarisi ve bu mimaride bulunan bileşenler açık bir şekilde görülmektedir.

FPGA iç mimarisindeki programlanabilir bileşenlerinden dolayı birçok alanda kullanımı cazip hale gelmiştir. Bundan dolayı, kullanım alanı oldukça fazla ve gereksinim duyan kullanıcı sayısı her geçen gün artmaktadır.




Şekil 6.1 : FPGA iç mimarisi ve içerdiği bileşenlerin görünümü.

Kullanım alanlarından bazıları şu şekilde sıralanabilir: otomotiv sektöründe, veri merkezlerinde, görüntü işleme projelerinde, tüketici elektroniği sektöründe, kablolu ve kablosuz haberleşme projelerinde ve yüksek performansla hesaplama ve veri depolama işlemlerinde oldukça fazla tercih edilmektedir.

FPGA üretici firmalarından en önemli ikisi Xilinx ve Altera firmalarıdır. Bu iki firmada değişik uygulamalara yönelik özel devreler üretmektedir. Tez çalışmamda Xilinx tarafından üretilmiş bir FPGA kullandığım için bu firmaya ait olan bazı modellerin sahip olduğu özellikler Çizelge 6.1'de listelenmiştir.

Çizelge 6.1 : Xilinx firmasına ait olan FPGA modellerinin kaynak durumları.

| **Model** | **Mantık Hücresi** | **Blok RAM (kB)** | **DSP hücresi** | **Girdi/Çıktı sayısı** |
|---|---|---|---|---|
| Spartan-7 XC7S100 | 102400 | 4320 | 160 | 400 |
| Artix-7 XC7A200T | 215360 | 13140 | 740 | 500 |
| Kintex-7 XC7K480T | 477760 | 34380 | 1920 | 400 |
| Kintex-7 XC7K410T | 406720 | 28620 | 1540 | 500 |
| Virtex-7 XC7VX690T | 693120 | 52920 | 3600 | 1000 |

Mantıksal kaynaklar, FPGA üzerinde bulunur ve mantıksal işlevleri gerçekleştirmek üzere kullanılmaktadırlar. Mantıksal kaynaklar, yapılandırılabilir mantık blokları oluşturmak için dilimler halinde gruplandırılmıştır. Bir dilim, belirli sayıda taramalı tablo (look up table), iki durumlu devre (flip-flop) ve çoklayıcı içerir. Bir LUT, FPGA'de kablolanmış mantık kapılarının bir koleksiyonudur. LUT'lar, her giriş kombinasyonu için önceden tanımlanmış bir çıktı listesini depolar ve bir mantık işleminin çıktısını almak için hızlı bir yol sağlar. İki durumlu devre, iki istikrarlı durumu olan ve tek bir biti temsil eden bir devredir. Bir çoklayıcı da, iki veya daha fazla girdi arasından seçim yapan ve seçilen girişi çıktılayan bir devredir.

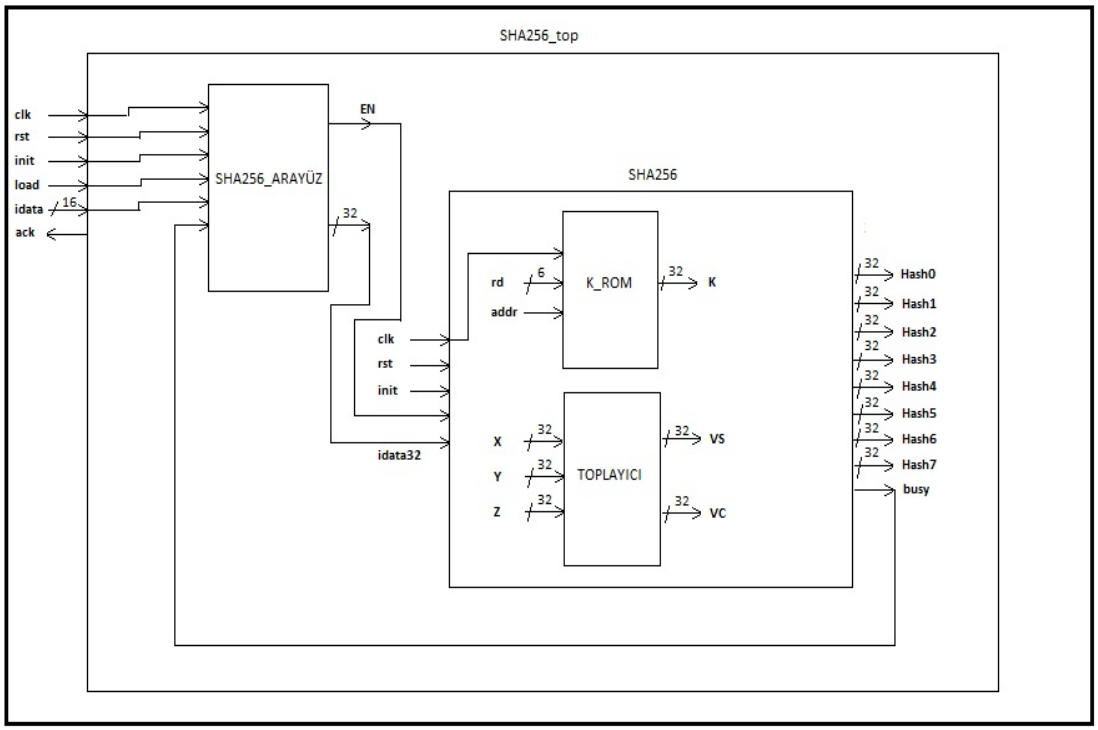
Farklı FPGA aileleri, dilimleri ve LUT'ları farklı şekilde uygulamaktadır. Örneğin, bir Virtex-2 FPGA üzerindeki bir dilimde iki LUT ve iki flip-flop bulunur; ancak Virtex-5 FPGA'deki bir dilim dört LUT'a ve dört flip-flopa sahiptir. Buna ek olarak, FPGA ailesine bağlı olarak bir LUT'ye girişlerin sayısı iki ila 6 arasında değişmektedir.

Tez çalışması kapsamında Xilinx firmasına ait ve Virtex-7 ailesinden XC7VX690T FPGA'i kullanılmıştır. Bu FPGA modelinin sahip olduğu kaynaklar Çizelge 6.1'de verilmiştir. Çalışmamızın temel konusu olan Bitcoin madenciliğinin ihtiyaç duyduğu algoritmalar Verilog programlama dilinde yazılmış olup yukarıdaki donanımda gerçeklenmiştir.

### 6.2 SHA256 Özet Algoritmasının FPGA'de gerçeklenmesi

SHA256 özet algoritmasının detayları 3. bölümde ele alınmıştır. Bu algoritmada yer alan işlemlerin Verilog programlama dili yardımı ile gerçeklenmesinin ayrıntılarını inceleyelim. Öncelikle SHA256 özet algoritmasının başlangıcında mesaj bloğunun 512 bitin katı olacak şekilde 0 ile genişletilmesi işlemi yapılmaktadır. Bitcoin madenciliğindeki blok başlığı değerine bakıldığında 640 bitten oluştuğu görülmektedir. Bundan dolayı, Eşitlik (3.1)'e göre 0 eklenecek bit sayısı 319 olarak bulunur. Blok başlığı değeri hep aynı veri uzunluğuna sahip olduğu için bu ekleme işlemi blok başlık değerinin bulunduğu modülde doğrudan yapılmıştır. Sonrasında ise mesajın, SHA256 özet algoritmasının girdi uzunluğu olan 512'lik bloklar haline getirilmesi gerekmektedir. Bitcoin başlık değerinin boyutu genişletme işleminden sonra 1024 değerine ulaşmaktadır ve her zaman iki mesaj bloğu oluşacağından verinin bulunduğu modülde adres değerinin kontrolü ile bu işlem

gerçekleştirilmektedir. Mesaj planlayıcısı işlemi ise, şu an için test amaçlı olarak bir arayüz modülünde ROM içerisinde (normal şartlarda gelen mesaj değeri UART veya USB gibi seri haberleşme standartları kullanılarak okunacaktır) yer alan mesaj değerlerinin ilgili kısımları SHA256 özet algoritmasının gerçeklendiği modüle beslenmekte ve burada Eşitlik (3.2-3.3) işlemleri yapılarak 512 bitten oluşan mesaj bloğundan 2048 bitlik bir mesaj değeri elde edilmektedir. Bu değer elde edilirken 17 adet 32 bitlik yazmaç kullanılmıştır ve her bir yazmaçtaki değer solundaki diğer yazmaca her saat darbesinde aktarılarak önceki değerlerin kullanılması gerekli olduğu durumlarda kolaylıkla erişim sağlanmıştır. Önceki değerine ihtiyaç duyulmayan değişkenler tel (wire) olarak tanımlanmıştır ve anlık değerleri kullanılarak işlemler gerçekleştirilmiştir. Şekil 6.2'de Verilog programlama dili yardımı ile gerçekleştirilmiş SHA256 özet algoritmasının blok diyagramı görülmektedir.




Şekil 6.2 : SHA256 özet algoritmasının detaylı mimarisi.

Şekil 6.2'de mesaj verisinin dışarıdan geldiği görülmektedir, fakat şimdi test amaçlı olarak mesaj verisi arayüz modülü içerisinde gömülü olarak bulunmaktadır ve bir adres değerine göre işlemde kullanılması gereken mesaj değeri okunmaktadır. Şekil 6.3'de arayüz modülünde yer alan mesaj verileri görülmektedir.

```
always @(posedge clk)
    begin
        case(i)
           7'b0000001: idata <= 16'h6162;
           7'b0000010: idata <= 16'h6380;
           7'b0000011: idata <= 16'h0000;
           7'b0000100: idata <= 16'h0000;
           7'b0000101: idata <= 16'h0000;
           7'b0000110: idata <= 16'h0000;
           7'b0000111: idata <= 16'h0000;
           7'b0001000: idata <= 16'h0000;
           7'b0001001: idata <= 16'h0000;
           7'b0001010: idata <= 16'h0000;
           7'b0001011: idata <= 16'h0000;
```

Şekil 6.3 : Arayüz modülünde yer alan mesaj verilerinin görünümü.

Ayrıca, arayüz modülünde SHA256 işlemlerinin gerçekleşme anını belirleyen izin sinyali (EN) üretilmektedir ve izin sinyalinin durumu özet algoritmasının temel işlemlerinin gerçeklendiği modülde tanımlanmış meşgul (busy) sinyaline ve arayüz modülünde yer alan sonlu durum makinesinin değerine göre değişmektedir. Arayüz modülünde yer alan sonlu durum makinesinin durumları 0, 1 ve 2 olarak değişmektedir. Eğer okuma yapılması gerekiyorsa durum makinesi 1 değerini almakta, özet algoritmasında bulunan işlemlerin yapılması gerektiğinde ise 2 değerini almaktadır. Mesaj verisinin okunması durumunda 32 bitlik değer alındıktan sonra değer ile ilgili yapılması gereken işlemlerden dolayı durum makinesi 2 değerini alır ve böylelikle izin sinyaline 1 değeri atanmış olur. Mesaj verisinin okuma işlemi tamamlandıktan sonra izin sinyalinin durumu meşgul sinyaline göre değişmektedir. Mesaj verisinin okunması durumunda meşgul sinyali 0 olarak tanımlanmış, okuma bittiğinde ise özet sonucu alınana kadar 1 değeri atanmıştır. Böylelikle izin sinyali mesaj verisinin okuması tamamlandığında 1 değerini almaktadır ve işlemler sona erene kadar bu durumda kalmaktadır. Şekil 6.4'de izin sinyalinin üretildiği sonlu durum makinesinin Verilog modeli görülmektedir.

```
assign EN = ((state == 3'b010) && (~Dnum[0]))? 1'b1 : 1'b0;
always @(load or state or busy) begin
   case (state)
      3'b000      : begin
         if (load) next_state <= 3'b001;
         else next_state <= 3'b000;
      end

      3'b001   : next_state <= 3'b010;

      3'b010   : begin
         if (~busy) next_state <= 3'b000;
         else next_state <= 3'b010;
      end

      3'b100   : next_state <= 3'b000;

      default: next_state <= 3'b000;
   endcase
```

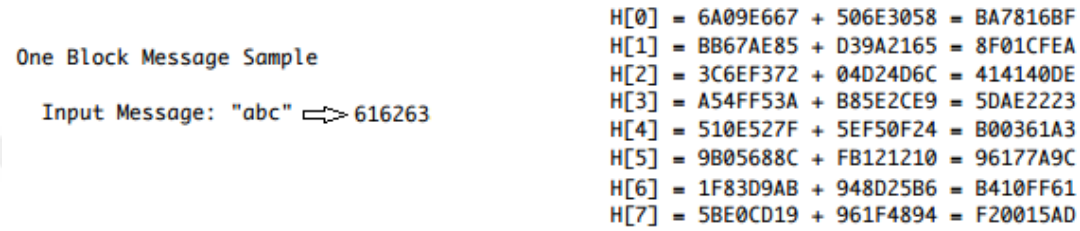
Şekil 6.4 : İzin sinyalinin üretildiği sonlu durum makinesinin Verilog modeli.

SHA256 modülünün içerisinde tanımlanmış ROM modülünde her bir döngü için kullanılan K adı verilen anahtar değerleri bulunmaktadır ve bu değerler ilk 64 asal sayının küp kökü alındığında elde edilen kesirli kısmın ilk 32 bitinden oluşmaktadır. Bu değerler, döngü sayısının durumuna göre ilgili adres değeri ve izin sinyalinden üretilen bir okuma sinyali yardımı ile elde edilmektedir. Ayrıca, özet algoritmasında yer alan bazı toplama işlemleri elde kaydetmeli toplayıcı ile gerçekleştirilmiştir. Bunun nedeni, Eşitlik (3.6) gibi bazı işlemlerin yapılabilmesi için 3 ve daha fazla 32 bitlik girdinin toplanması gerekmektedir ve bu işlem 3 girdiyi aynı anda alan elde kaydetmeli toplayıcı ile gerçekleştirilmiştir, böylelikle daha verimli şekilde sonuca ulaşılmaktadır. Şekil 6.5'de elde kaydetmeli toplayıcının Verilog modeli görülmektedir.

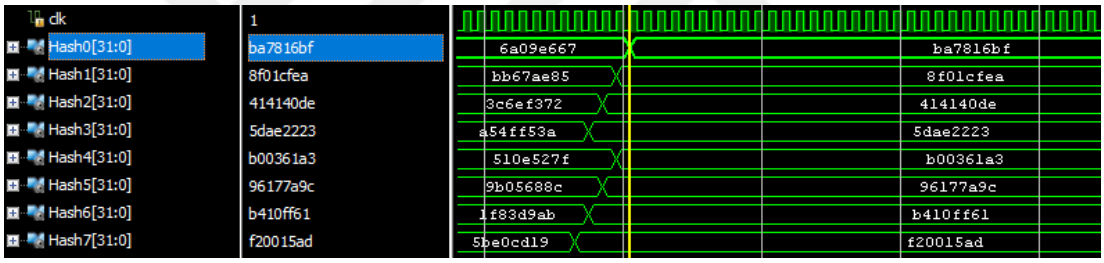
```
`timescale 1ns / 1ps
module CSA(
   input [31:0] X,
   input [31:0] Y,
   input [31:0] Z,
   output [31:0] VS,
   output [31:0] VC
);

assign VS = X ^ Y ^ Z;
assign VC = {((X[30:0] & Y[30:0]) | ((X[30:0] ^ Y[30:0]) & Z[30:0])),1'b0};

endmodule
```

Şekil 6.5 : Elde kaydetmeli toplayıcının Verilog modeli.

Yapılan gerçeklemenin doğruluğunu test etmek için Ulusal Standartlar ve Teknoloji Enstitüsü (NIST) tarafından yayınlanmış olan SHA256 özet algoritması için örnek girdi ve çıktı değerleri referans alınmıştır. Verilog modeli ile gerçekleştirdiğimiz algoritmanın simülasyonları ve FPGA gerçeklemesi sonucu NIST tarafından verilmiş referans değeri ile karşılaştırılarak sistemin doğruluğu test edilmiştir. Şekil 6.6'da NIST tarafından paylaşılmış SHA256 özet algoritmasının sonucu, Şekil 6.7'de ise Verilog modelinin FPGA gerçeklemesi sonuçları yer almaktadır.




Şekil 6.6 : NIST tarafından yayınlanmış referans değerleri [13].




Şekil 6.7 : Verilog modelinin FPGA gerçeklemesi sonuçları.

### 6.3 Bitcoin Madenciliği Sisteminin FPGA'de gerçeklenmesi

Bir önceki bölümde SHA256 özet algoritmasının detaylarından bahsedilmiştir. Bitcoin madenciliği sisteminin de temel işlemi SHA256 özet fonksiyonudur. Bitcoin madencileri sistem tarafından verilen ödülü kazanmak için yeni blokları doğrulamaya çalışmaktadırlar. Bunun için de üç SHA256 algoritması kullanılmaktadır. Şekil 6.2'de verilen SHA256 özet algoritmasının detaylı mimarisinden farklı olarak SHA256_top modülünde bu üç özet işleminin ardarda yapılabilmesi için gerekli kontrol sinyalleri üretilmiştir ve sistemin seri bir şekilde çalışması sağlanmıştır. Ayrıca, 1. SHA256 işleminden sonra elde edilen değer 2. SHA256 işleminin başlatılması için kullanılmaktadır. Bundan dolayı SHA256_top modülünde iv değişkenleri tanımlanmış ve 1. ve 3. SHA256 özet fonksiyonu için başlatma vektörü değerlerinin ataması yapılmıştır, ancak 2. SHA256 özet fonksiyonu için 1. SHA256

özet fonksiyonunun çıktıları beslenmiştir. Şekil 6.8'de iv değişkenine yapılan atamalar görülmektedir.

```
        if(rst)begin
            iv0_r <= 32'h00000000;
            iv1_r <= 32'h00000000;
            iv2_r <= 32'h00000000;
            iv3_r <= 32'h00000000;
            iv4_r <= 32'h00000000;
            iv5_r <= 32'h00000000;
            iv6_r <= 32'h00000000;
            iv7_r <= 32'h00000000;
         end
         else if(out_delay && (control == 2'b10))begin
            iv0_r <= Hash0;
            iv1_r <= Hash1;
            iv2_r <= Hash2;
            iv3_r <= Hash3;
            iv4_r <= Hash4;
            iv5_r <= Hash5;
            iv6_r <= Hash6;
            iv7_r <= Hash7;
         end
    end

    assign iv0 = (init) ? 32'h6a09e667 : (iv_control) ? iv0_r : 32'h6a09e667;
    assign iv1 = (init) ? 32'hbb67ae85 : (iv_control) ? iv1_r : 32'hbb67ae85;
    assign iv2 = (init) ? 32'h3c6ef372 : (iv_control) ? iv2_r : 32'h3c6ef372;
    assign iv3 = (init) ? 32'ha54ff53a : (iv_control) ? iv3_r : 32'ha54ff53a;
    assign iv4 = (init) ? 32'h510e527f : (iv_control) ? iv4_r : 32'h510e527f;
    assign iv5 = (init) ? 32'h9b05688c : (iv_control) ? iv5_r : 32'h9b05688c;
    assign iv6 = (init) ? 32'h1f83d9ab : (iv_control) ? iv6_r : 32'h1f83d9ab;
    assign iv7 = (init) ? 32'h5be0cd19 : (iv_control) ? iv7_r : 32'h5be0cd19;
```

Şekil 6.8 : IV değişkenine yapılan atamalar.

Ayrıca 2. SHA256 özet fonksiyonundan elde edilen değer 3. SHA256 özet fonksiyonuna girdi olarak beslenmelidir. Bundan dolayı 2. özet fonksiyonunun sonucu arayüz modülüne gönderilerek ilgili verilerin beslenmesi gerçekleştirilir. Bunlara ek olarak, 3. SHA256 özet fonksiyonunun çıktısının hedef değerinden küçük olup olmadığına bakılmaktadır. Bundan dolayı, sistem tarafından verilen 32 bitlik hedef değerinden 256 bitlik hedef değerinin elde edilmesi için bir modelleme yapılmıştır. Şekil 6.9'da hedef modülünün görünümü yer almaktadır.

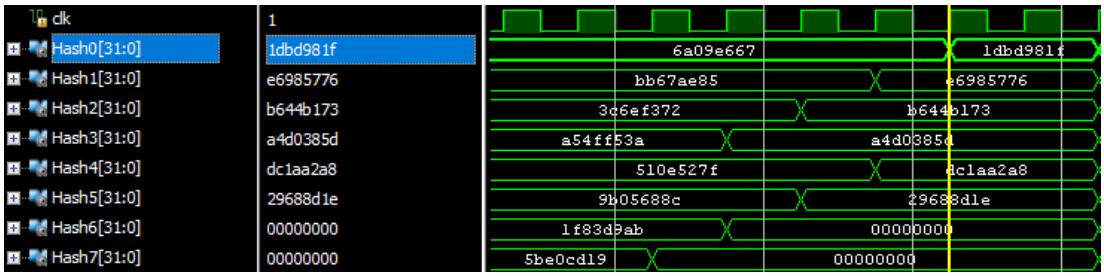
```
module target(
    input clk,
    input rst,
    input [31:0] target32,
    output reg [255:0] target256
    );

    reg [7:0] exponent;
    reg [23:0] mantissa;

    always @(posedge clk)begin
        if(rst)begin
            exponent <= 1;
            mantissa <= 1;
        end
        else if((target32 > 0) && (exponent == 1))begin
            exponent <= target32[31:24] - 2'h03;
            mantissa <= target32[23:0];
        end
    end

    always @(posedge clk)begin
        if(rst)
            target256 <= 1;
        else if((target32 > 0) && (exponent > 1) && (target256 == 1))begin
            target256 <= {target256[255:24],mantissa[23:0]};
        end
        else if((target32 > 0) && (exponent > 1) && (target256 == mantissa))begin
            target256 <= target256 << (exponent*8);
        end
```

Şekil 6.9 : Hedef modülünün görünümü.

Yapılan uygulamanın doğruluğu, internetteki bir kaynakta yer alan blok başlığı değerine karşılık elde edilmesi gereken özet değerine göre test edilmiştir. Şekil 6.10'da kaynakta yer alan değerler ve Şekil 6.11'de de gerçekleştirilen Bitcoin madencilik sisteminin FPGA sonuçları yer almaktadır.

```
>>> import hashlib
>>> header_hex = ("01000000" +
 "81cd02ab7e569e8bcd9317e2fe99f2de44d49ab2b8851ba4a308000000000000" +
 "e320b6c2fffc8d750423db8b1eb942ae710e951ed797f7affc8892b0f1fc122b" +
 "c7f5d74d" +
 "f2b9441a" +
 "42a14695")
>>> header_bin = header_hex.decode('hex')
>>> hash = hashlib.sha256(hashlib.sha256(header_bin).digest()).digest()
>>> hash.encode('hex_codec')
'1dbd981fe6985776b644b173a4d0385ddc1aa2a829688d1e0000000000000000'
>>> hash[::-1].encode('hex_codec')
'0000000000000001e8d6829a8a21adc5d38d0a473b144b6765798e61f98bd1d'
```

Şekil 6.10 : Gerçekleştirilen sistemin doğrulanması için kullanılan verilerin görünümü [14].




Şekil 6.11 : Gerçekleştirilen Bitcoin madencilik sisteminin FPGA sonuçları.

## 6.4 Literatürde önerilmiş olan iyileştirme tekniklerinin FPGA'de gerçeklenmesi

5. bölümde detayları açıklanan iyileştirme tekniklerinin FPGA'de gerçeklenmesinin performans da meydana getirdiği katkı ve bu iyileştirme yöntemleri uygulandığında kaynak tüketiminde nasıl bir değişime neden olduğu incelenecektir.

**- $SHA256_0$ için H0 hesaplanması**

Bu tür bir optimizasyon günümüzdeki madencilik aygıtlarında kullanılmaktadır. Bu yüzden performans değeri bulunurken bu optimizasyon hesaba katılmayacaktır. Ancak, iyileştirme yönteminin uygulanmasında bazı kontrol sinyalleri kullanılmaktadır. 2 bitlik bir kontrol sinyali başlangıçta 0 olarak tanımlanmıştır ve bu değer 0'dan farklı olduğunda $SHA256_0$ işleminin çalışması engellenmektedir. Ayrıca arayüz modülünde yer alan blok başlığı değerinin ilk 32 adresi başlangıçta okunduktan sonra bir daha okunması engellenerek $SHA256_0$'a ait olan verilerin sisteme tekrar beslenmesi engellenmektedir.

**- $SHA256_2$'de önceden özet değerinin kontrolünün yapılması**

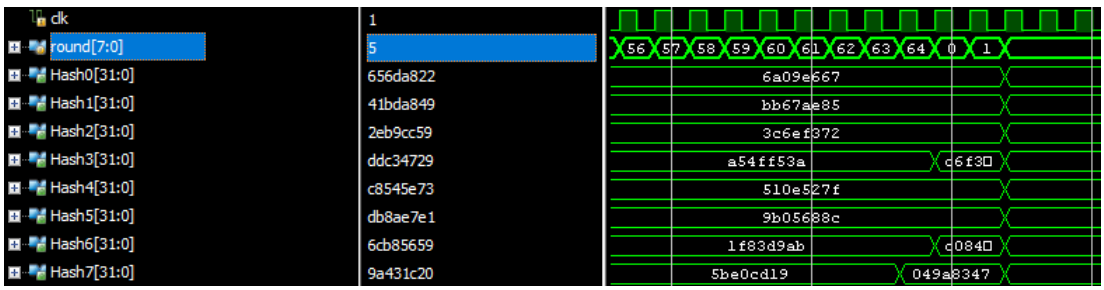
5. bölümde belirtildiği gibi 64. döngü sonucunda elde edilen G ve H çalışma değişkenlerine 61. ve 62. döngüdeki E çalışma değişkeninden ulaşılmaktadır. Ayrıca hedef değeri, 2016 bloğun bulunmasından sonra iş kanıtı problemini zorlaştırmak için küçültülmektedir. Bundan dolayı, günümüzde hedef değerinin ilk 64 biti 0'dan oluşmaktadır. Böylelikle 61. ve 62. döngüdeki E çalışma değişkeninin başlatma vektöründeki ilgili değer ile toplanması sonucu 64 bit 0 elde edilmesi gerekmektedir. Aksi takdirde iş kanıtı probleminin çözülemediği ve nonce değerinin artırılarak denemelerin devam edilmesi gerektiği anlaşılmaktadır. Gerçeklediğimiz sistemde 64 döngüden oluşan SHA256 özet algoritması, mesaj verilerinin hazırlanabilmesi için 67 döngüden oluşturulmuştur. Bundan dolayı önceden kontrol etme işlemi 64 ve 65.

döngüde gerçekleştirilmiştir. SHA256 modülünde bu kontrol işleminin yapıldığı kısmın görünümü Şekil 6.12'de yer almaktadır.

```
always @(posedge clk)begin
    if (rst) round <= 0;
   else begin
      if (init) round <= constant - 1;
      else if (EN) begin
         if(out) round <= 0;
         else if (round < 'd67) round <= round + 1;
         else if (round == 'd67) round <= 0;
         else round <= round;
      end
    else round <= round;
   end
end

always @(posedge clk)begin
    if(rst) out <= 0;
    else if(round == 'd66) out <= 1;
    else if((compare_control == 2'b10) && (round == 63) && (Hash7 + e != 32'h00000000)) out <= 1;
    else if((compare_control == 2'b10) && (round == 64) && (Hash6 + e != 32'h00000000) && ~out) out <= 1;
    else out <= 0;
end
```

Şekil 6.12 : $SHA256_2$ özet değerinin önceden kontrolünün yapılması.

Şekil 6.12'de görüldüğü gibi özet değeri önceden kontrol edilmiştir ve eğer 64 bitlik ilk kısmı 0'a eşit değilse SHA256 modülünde işlemin bittiğini gösteren out sinyaline 1 ataması yapılmaktadır ve bu değer SHA256_top modülünde tekrar özet fonksiyonunun çalışmaya başlaması için SHA256 modülüne init sinyali olarak beslenmektedir. Gerçekleştirlen iyileştirme tekniğinin FPGA sonuçları Şekil 6.13'da yer almaktadır.

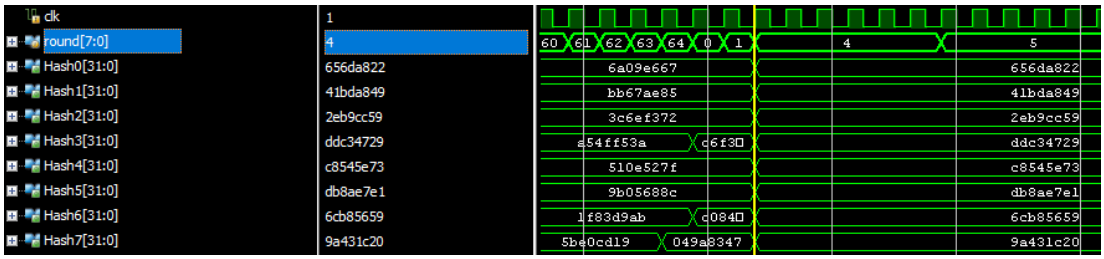



Şekil 6.13 : $SHA256_2$'de önceden özet değerinin kontrolünün yapılması ile elde edilen FPGA sonuçlarının görünümü.

**- $SHA256_1$'in ilk üç döngüsü**

$SHA256_1$'in mesaj verisinin ilk üç 32 bit değeri sırası ile Merkle kökü değerinin son 32 biti, zaman damgası ve hedef değeridir. Bu değerler, iş kanıtı problemini çözmek için kullanılan nonce değerinin $2^{32}$ denemesi sona erene kadar değişmemektedir. Bundan dolayı, bu ilk üç döngü bir kez yapıldıktan sonra ilgili değerler yazmaçlarda tutulmuştur ve bu değerler $SHA256_1$ tekrar çalıştırıldığında başlangıç değerleri

olarak beslenmiştir. Böylelikle, başlangıçta 4. döngüden başlanmaktadır ve 3 döngülük bir kazanç sağlanmıştır. Gerçekleştirilen iyileştirme tekniğinin FPGA sonuçları Şekil 6.14'de yer almaktadır.




Şekil 6.14 : $SHA256_1$'in ilk üç döngüsü için yapılmış olan iyileştirme yöntemi sonucunda elde edilen FPGA sonuçlarının görünümü.

**- $SHA256_1$'in 4. döngüsünün artan olarak hesaplanması**

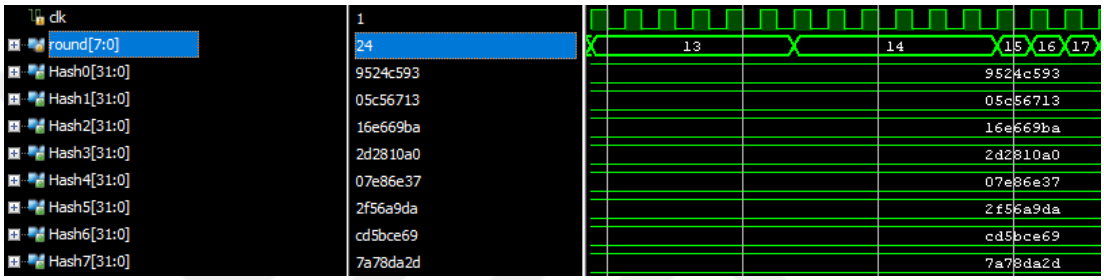
$SHA256_1$ özet fonksiyonuna beslenen dördüncü 32 bitlik mesaj nonce değeridir ve bu değer iş kanıtı problemi çözülene kadar 1 artırılır. Bundan dolayı 8 adet çalışma değişkeninde E ve A değerleri her zaman 1 artmaktadır, diğer değerler ise ilk kez 4. döngü yapıldığında hangi değere sahipseler yine aynı değere sahip olmaya devam etmektedirler. Bundan dolayı, bu değerlerde iv_opt adı verilen yazmaçlarda saklanmaktadır ve $SHA256_1$ özet fonksiyonu bu değerlerin beslenmesi ile başlamaktadır. Böylelikle özet fonksiyonundaki 1 döngülük işlemin yapılması ortadan kaldırılmıştır. Şekil 6.15'de ilgili değerlerin yazmaçlara kaydedilmesi görülmektedir ve Şekil 6.14'de uygulanan iyileştirme tekniğinin FPGA sonuçları görülmektedir.

```
always @(posedge clk)begin
    if(rst)
        iv0_opt <= 32'h00000000;
    else if(iv_control && control == 2'b10 && i==47)
        iv0_opt <= a + 32'h01000000;
    else if(iv_control && control == 2'b00 && i==46)
        iv0_opt <= a + 32'h01000000;
end

always @(posedge clk)begin
    if(rst)
        iv4_opt <= 32'h00000000;
    else if(iv_control && control == 2'b10 && i==45)
        iv4_opt <= e + 32'h01000000;
    else if(iv_control && control == 2'b00 && i==44)
        iv4_opt <= e + 32'h01000000;
end
```

Şekil 6.15 : $SHA256_1$ başlatılırken kullanılacak değerlerin artan olarak hesaplanması.

**- $SHA256_1$ ve $SHA256_2$'nin genişletilmiş mesaj bloklarından bazılarının 0 olması ve mesaj uzunluğun sabit olması**

5. bölümde yer alan son iki iyileştirme yöntemi birarada gerçekleştirilmiştir. SHA256 özet algoritmasında öncelikle mesaj değerlerinin gelmesi beklenmekte ve sonrasında ilgili işlemler yapılmaktadır. Ancak mesaj bloklarından bazılarının 0 olması ve $SHA256_1$'de mesaj uzunluğunun 640, $SHA256_2$'de mesaj uzunluğunun 256 olduğu bilindiğinden mesaj verisi için herhangi bir bekleme yapılmadan işlemlere başlanmaktadır. Böylelikle özet fonksiyonunda bir döngülük iyileştirme sağlanmıştır. Şekil 6.16'da elde edilen iyileştirme sonucu görülmektedir.




Şekil 6.16 : Mesaj bloklarından bazılarının 0 olması ve mesaj uzunluğunun sabit olması sonucunda elde edilen FPGA sonuçlarının görünümü.

İyileştirme yöntemlerinin FPGA'de uygulanmasından sonra performansda meydana gelen değişim ve kaynak tüketimi ile güç tüketimi değerleri şu şekildedir. Yapılan iyileştirmeler sonucunda 30 saat darbelik bir performans artışı sağlanmıştır. İyileştirme yapılmamış sistemde iş kanıtı probleminin çözümü için kullanılan iki SHA256 fonksiyonu 298 saat darbesi, iyileştirme yapılmış sistem ise 268 saat darbesi sürmektedir. Buradan performans artışı şu şekilde bulunur:

Performans Artışı = Elde edilen saat darbe kazancı / Normal gerçeklemenin saat darbe sayısı = 30 / 298 = %10.1

Bu performans artışının neden olduğu maliyete ise kaynak tüketimi ve güç tüketimi değerlerinden ulaşılabilir. Güç tüketimi değerlerini incelediğimizde; iyileştirme tekniklerinin uygulanmadığı sistemde güç tüketiminin 0.337 W olduğu sonucuna ulaşılmıştır. İyileştirme tekniklerinin uygulandığı sistemde ise güç tüketimi 0.335 W olarak ortaya çıkmıştır. Çizelge 6.2'de iyileştirilmiş ve herhangi bir optimizasyon uygulanmamış sistemlerin kaynak tüketimi değerleri bulunmaktadır.

Çizelge 6.2 : İyileştirilmiş ve herhangi bir optimizasyon uygulanmamış sistemlerin kaynak tüketimi değerlerinin karşılaştırılması.

| Kaynak | İyileştirilmiş Sistem | Standart Sistem |
|---|---|---|
| | Tüketim | Tüketim |
| **LUT** | 4586 | 4447 |
| **LUTRAM** | 370 | 370 |
| **FF** | 5284 | 4478 |
| **BRAM** | 7.50 | 7.50 |

Kaynak tüketimi değerlerine bakıldığında iki durumlu devre (Flip Flop) bileşeninde en çok artış meydana geldiği görülmektedir. Ancak, FPGA daha az sayıda taramalı tablo (Look up table) içermektedir. Bundan dolayı sınırlayıcı bileşen taramalı tablo kaynağıdır. Çizelge 6.2'deki değerlere bakıldığında taramalı tablo kaynak tüketiminde %3'lük bir artış olduğu görülmektedir. Kaynak tüketiminin performansa negatif bir etkisi olduğu için %3'lük bir düşüş meydana gelmektedir. Böylelikle yapılan iyileştirme sonucunda yaklaşık olarak %7'lik bir performans artışı sağlanmıştır. Uygulanan iyileştirme yöntemlerinin performans artışının yanında çok büyük maliyetlere neden olmadığı görülmektedir. Bu nedenle yapılan iyileştirmenin Bitcoin madenciliği için uygulanabilir olduğu ve özet üretme hızında ciddi artışlar meydana geleceği görülmektedir. Ayrıca, Bitcoin madenciliğinde oluşan çok büyük elektrik tüketimi değeri de önemli ölçüde azaltılacaktır. 2016 yılında Bitcoin madenciliği için harcanan elektrik tüketimi 400 milyon $ olarak kaynaklarda geçmektedir [15]. Böylelikle yapacağımız iyileştirme teknikleri ile yaklaşık olarak 28 milyon $ tasarruf edilebilir.

# 7. İLGİLİ ÇALIŞMALAR

Yüksek hızda SHA256 algoritmasını çalıştırmanın yolu FPGA ya da ASIC gibi donanım tabanında kodlama ile gerçekleştirilir. Bitcoin madenciliği için artık CPU ve GPU'nun kullanımı söz konusu değildir. Bu tür aygıtlar ile donanım tabanlı uygulamaların özet hızına yetişmek imkansızdır. Bu tür donanımsal uygulamalar tamamen yüksek hız için hizmet vermektedir ve bazı yapılan donanımsal optimizasyonlar ile verimlilik daha da artırılabilir ve güç tüketimi azaltılabilir. Bitcoin madenciliğine yönelik yapılmış olan optimizasyonlar, genel SHA256 algoritmasının donanımsal uygulamasını hızlandırma amacıyla yapılmıştır ve Bitcoin madenciliğine özgü bir yöntem sadece bir çalışmada önerilmiştir. Donanımsal optimizasyonlar ile daha çok SHA256 algoritmasında yer alan Eşitlik (3.11)'deki 7 toplama işleminde iyileştirme yapılması amaçlanmıştır. Aşağıda SHA256 algoritmasının donanım tabanında hızlandırılması için önerilmiş olan çözüm yöntemleri ele alınacaktır.

SHA256 özet algoritmasının birçok donanım tabanlı uygulaması literatürde bulunmaktadır. Bu uygulamalar, FPGA ya da ASIC tasarımlarını gerçekleştirmiştir. Bu uygulama tasarımları SHA256 algoritmasının verimliliğini arttırmak amacıyla aşağıda belirtilen optimizasyonlardan birini veya birkaçının birleşimini içermektedir [16-22].

Yukarıda bahsedildiği gibi SHA256 algoritmasında Eşitlik (3.11)'de verilen a değişkeninin hesaplanmasında uzun veri yolu bulunmaktadır. Yani, a değişkeni 7 değerin mod $2^{32}$'de toplanması ile elde edilmektedir. Bu sorunun çözümü için önerilen mimari, toplam ve elde yollarının birbirinden ayrılarak elde yayılımının neden olduğu gecikme, elde kaydetme toplayıcısı ile minimize edilir [23]. Elde kaydetme toplayıcısı 3 işleci girdi olarak alır ve a değişkeninin hesaplanması için sadece 5 adet elde kaydetme toplayıcısı kullanılır.

SHA256 algoritmasının sıkıştırma fonksiyonunda birden fazla döngünün birleşimsel mantık kullanarak aynı anda uygulanması sağlanabilir ve bu sayede SHA256 algoritması için gerekli olan saat çevrimi sayısı düşürülür [24]. Bu tür optimizasyon

mesaj sıkıştırma fonksiyonunda yer alan verilere bağlı olarak verimliliğin artışına yardımcı olur. Bu tür bir optimizasyon ile saat çevrimi sayısının azalması sağlanırken kaynak tüketiminde ciddi artışlar meydana gelebilmektedir.

Özet Fonksiyonunda Önerilen Optimizasyonlar başlığı altında belirttiğimiz genişletilmiş mesaj bloklarından bazılarının 0 olması ve toplama işleminin azaltılması yöntemine benzer şekilde bazı çalışmalarda $K_j + W_j$ toplamı başlangıçta yapılmıştır [24]. Bunun yapılmasında bir sakınca yoktur, çünkü $K_j$ ve $W_j$ değişkenleri diğer değerlere göre önceden bilinmektedir ve birbirlerinden bağımsızdırlar.

Literatür taraması sonucu, Bitcoin madenciliğine özgü bir optimizasyon çalışmasının çok az sayıda olduğu görülmekte ve bu çalışmalarda da önerilen yöntemlerin herhangi bir uygulaması söz konusu değildir ve performans artışı bakımından bir sonuç elde edilmemiştir [11]. Bu nedenle bu çalışma ile literatürde önerilen yöntemleri donanım tabanlı olarak gerçekleştirmeyi ve performans bakımından oluşan artış miktarı hakkında kesin bir değer vermeyi sağlamış olduğumuzu söyleyebiliriz.

# 8. SONUÇ

Yaptığımız çalışmanın en önemli amacı SHA256 özet algoritmasında, Bitcoin madenciliği için spesifik olan durumlar ele alınarak optimizasyon metodlarının uygulanmasıdır. Birçok donanım tabanlı optimizasyon SHA256 özet algoritmasının donanım uygulamalarında kullanılmıştır. Fakat bu optimizasyonlar, SHA256 fonksiyonunun genel durumu için üretilmiştir. Bundan dolayı Bitcoin madenciliğine özgü çözümlerin eksik olduğu düşünülmüş ve bu konu üzerinde araştırmalar yapılarak bir takım optimizasyon yöntemlerine ulaşılmıştır.

Yaptığımız çalışmanın temel katkısı, Bitcoin madenciliğine özgü SHA256 özet fonksiyonunda incelenen optimizasyon yöntemlerinin performansa olan etkisini ortaya koyma ve bu çözümlerin donanım tabanında kullanılabilirliğinin gösterilmesidir. Burada yapılan eniyilemeler genel SHA256 algoritmasında bir hızlandırma yapmamasına rağmen Bitcoin madenciliği için önemli gelişmeler sağlamıştır. Literatürde önerilmiş olan iyileştirme teknikleri FPGA'de gerçekleştirilmiştir. Eğer önerilen optimizasyon işlemleri madencilik aygıtlarında uygulanırsa, iki SHA256 özet fonksiyonunun işlem süresi yaklaşık olarak 1.8611 SHA256 işlem süresine düşmektedir. Aynı zamanda, Bitcoin madenciliğinde en önemli problem yüksek miktarda elektrik tüketiminin olmasıdır. 2016 yılı verilerine bakıldığında Bitcoin madenciliği için yaklaşık olarak 400 milyon $ elektrik tüketimi gerçekleşmektedir, önerilen yöntemlerin uygulanması sonucunda yaklaşık olarak 28 milyon $ değerinde tasarruf elde edilecektir. Şu an için Bitcoin madenciliğinin büyük oranda ASIC aygıtlarında yapıldığı düşünüldüğünde FPGA'de yapılan kodlama ASIC'lerde kullanılabilir ve ciddi performans artışı sağlanabilir.

## KAYNAKLAR


[1] **Antonopoulos, A.M.,** *Mastering Bitcoin*, O'Reilly Media, USA, 2014.

[2] http://bitcoincharts.com/, alındığı tarih: 10.08.2016.

[3] http://www.economist.com/news/finance-and-economics/mining-digital, alındığı tarih: 15.08.2016.

[4] https://blockchain.info/stats, alındığı tarih: 10.08.2016.

[5] **Nakamoto, S.,** Bitcoin - A Peer-to-Peer Electronic Cash System, https://bitcoin.org/bitcoin.pdf, alındığı tarihi: 10.08.2016.

[6] http://bitcoinfographics.com/en/transaction-life-cycle/, alındığı tarih: 20.01.2016.

[7] ANSI, X9.62: 2005: The Elliptic Curve Digital Signature Algorithm (ECDSA), Public Key Cryptography for the Financial Services Industry, 2005.

[8] **Johnson, D., Menezes, A., Vanstone, S.,** (2001). The Elliptic Curve Digital Signature Algorithm (ECDSA), *International Journal of Information Security*, 1, 36-63.

[9] **Esuruoso, O. (2011),** *High Speed FPGA Implementation of Cryptographic Hash Function* (doctoral dissertation).

[10] National Institute of Standarts and Technology (NIST), SHA256 Standart, http://csrc.nist.gov/publications/fips/fips180-2pdf, alındığı tarih: 10.08.2016.

[11] **Courtois, N.T., Grajek, M., Naik, R.,** (2014) Optimizing SHA256 in Bitcoin Mining, *International Conference on Cryptography and Security Systems*, Lublin, Poland, September 22-24.

[12] https://en.bitcoin.it/wiki/Protocol_specification, alındığı tarihi: 10.08.2016.

[13] NIST, SHA-256 Example, http://csrc.nist.gov/groups/ST/toolkit/documents/Examples/SHA256.pdf, alındığı tarihi: 10.08.2016.

[14] https://en.bitcoin.it/wiki/Block_hashing_algorithm, alındığı tarih: 27.09.2016.

[15] https://www.cryptocoinsnews.com/, alındığı tarih: 27.01.2016.

[16] **Chaves, R., Kuzmanov, G., Sousa, L., Vassiliadis, S.,** (2008). Cost-Efficient SHA Hardware Accelerators, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 16, 999-1008.

[17] **Chaves, R., Kuzmanov, G., Sousa, L., Vassiliadis, S.,** (2006) Improving SHA-2 Hardware Implementations, *International Conference on Cryptographic Hardware and Embedded Systems*, Yokohama, Japan, October 10-13.

[18] **Dadda, L., Macchetti, M., Owen, J.,** (2004) An ASIC Design for a High Speed Implementation of the Hash Function SHA-256 (384, 512), *14th ACM Great Lakes Symposium on VLSI*, Boston, USA, April 26-28.

[19] **Dadda, L., Macchetti, M., Owen, J.,** (2004) The Design of a High Speed ASIC Unit for the Hash Function SHA-256 (384, 512), *Design, Automation and Test in Europe Conference and Exhibition*, Paris, France, February 16-20.

[20] **Grembowski, T., Lien, R., Gaj, K., Nguyen, N., Bellows, P., Flidr, J., Lehman, T., Schott. B.,** (2002) Comparative analysis of the hardware implementations of hash functions SHA-1 and SHA-512, *International Conference on Information Security*, Sao Paulo, Brazil, October 1-3.

[21] **Lien, R., Grembowski, T., Gaj, K.,** (2004) A 1 Gbit/s partially unrolled architecture of hash functions SHA-1 and SHA-512, *Cryptographers' Track at the RSA Conference*, San Francisco, USA, February 23-27..

[22] **Macchetti, M., Dadda, L.,** (2005) Quasi-Pipelined Hash Circuits, *17th IEEE Symposium on Computer Arithmetic*, Massachusetts, USA, June 27-29.

[23] **Crowe, F., Daly, A., Kerins, T., Marnane, W.,** (2004) Single-Chip FPGA Implementation of A Cryptographic Co-Processor, *IEEE International Conference*, Brisbane, Australia, December 6-8.

[24] **Ting, K. K., Yuen, S. C. L., Lee, K. H., Leong, P. H. W.,** (2002) An FPGA Based SHA-256 Processor, *International Conference on Field Programmable Logic and Applications*, Montpellier, France, September 2-4.

# ÖZGEÇMİŞ


**Ad-Soyad** : **Erşen BALCISOY**

**Uyruğu** : **T.C.**

**Doğum Tarihi ve Yeri** : **05.08.1989 / ANKARA**

**E-posta** : **ebalcisoy@gmail.com**


**ÖĞRENİM DURUMU:**

- **Lisans** : 2013, Ankara Üniversitesi, Mühendislik Fakültesi, Kimya Mühendisliği Bölümü
- **Yükseklisans** : 2017, TOBB Ekonomi ve Teknoloji Üniversitesi, Bilgisayar Mühendisliği Anabilim Dalı, Bilgisayar Mühendisliği

**MESLEKİ DENEYİM VE ÖDÜLLER:**

| Yıl | Yer | Görev |
|---|---|---|
| 2014 - … | Tera-Hz Mikroelekteronik Ltd.Şti | Mühendis |
| 2014 – 2017 | TOBB ETÜ | Araştırma Burslu Yük. Lis. Öğrencisi |

**YABANCI DİL: İngilizce**

**TEZDEN TÜRETİLEN YAYINLAR, SUNUMLAR VE PATENTLER:**

- **Balcısoy, E.,** Bıçakcı, K., 2016. Yüksek Performanslı Bitcoin Madencilliği için SHA256 Özet Algoritmasının Eniyilenmesi, ISC2016, October 25-26, Ankara, Turkey.

**DİĞER YAYINLAR, SUNUMLAR VE PATENTLER:**

- Çağlan, A., İnceöz, E., **Balcısoy, E.,** Özbek, M. E., Çavuş, E., 2016. FPGA Implementation of AWGN Noise Generator Using Box-Muller Method, 24th IEEE Signal Processing and Communications Applications Conference, May 16-19, Zonguldak, Turkey.