İSTANBUL TEKNİK ÜNİVERSİTESİ
ELEKTRİK-ELEKTRONİK FAKÜLTESİ

REED-SOLOMON KOD ÇÖZÜCÜ TASARIMI

BİTİRME ÖDEVİ

GİZEM KAÇMAZ
040090427

Bölümü: Elektronik ve Haberleşme Mühendisliği
Programı: Elektronik Mühendisliği

Danışmanı: Doç. Dr. Sıddıka Berna Örs Yalçın

MAYIS 2013

ÖNSÖZ

Bitirme çalışmam boyunca sonsuz anlayış ve sabırla bana yol gösteren, yardımını hiçbir zaman esirgemeyen danışman hocam Sayın Doç. Dr. Sıddıka Berna Örs Yalçın'a, değerli katkılarından dolayı Seyyid M. Dilek'e ve hayatım boyunca maddi ve manevi desteklerini hep hissettiğim aileme içten saygı ve teşekkürlerimi sunarım.

Mayıs 2013 Gizem Kaçmaz

İÇİNDEKİLER

KISALTMALAR

ECC **:**Error Correcting Coding
RS **:**Reed-Solomon
BCH **:**Bose-Chaudhuri-Hocquenghem
VHDL **:**Very High Speed Integrated Circuits Hardware Description Language
GF **:**Galois Field
LCM **:**Least Common Multiple
KES **:**Key Equation Solver
RTL **:**Register Transfer Level
FSM **:**Finite State Machine
ME **:**Modified Euclid
BM **:**Berlekamp-Massey
RiBM **:**Reformulated inversionless Berlekamp-Massey

REED SOLOMON KOD ÇÖZÜCÜ TASARIMI

ÖZET

Kod çözücü devreleri, girişindeki kod sözcüklerini, kodlanmış durumdaki çıkışlara dönüştüren sayısal devrelerdir. Girişler ve çıkışlar birbirinden farklı şekillerde kodlanmıştır ve kod çözücü devresinin işlevi kodlanmış verinin ilk halini elde etmektir. Kod çözücüler, veri çoğullama işlemlerinde, 7 parçalı göstergelerde ve bellek adreslerinin kod çözümlerinde yaygın olarak kullanılmaktadır.

Bir iletişim kanalına giren veriye kanal içinde eklenen gürültülerin belirlenerek verinin kanal çıkışında bu gürültülerden arındırılması işlemine hata düzeltme kodlaması (Error Correcting Coding-ECC) denir. Reed–Solomon(RS) kodlar, bir hata düzeltme kodu olmakla birlikte ikili olmayan bir Bose-Chaudhuri-Hocquenghem (BCH) kodudur. Bu tezde Reed- Solomon kodlar ve dahil olduğu BCH kodları ile ilgili bilgi verilmiştir.

Bu bitirme çalışması içeriğinde Reed–Solomon kodları ve özellikleri göz önünde bulundurularak RS kod çözücü devresi ele alınmış ve Çok Yüksek Hızlı Tümleşik Devreler Donanım Tanımlama Dili (Very High Speed Integrated Circuits Hardware Description Language–VHDL) kullanılarak gerçeklemesi yapılmıştır.

Gerçekleme aşamasında, tasarım için gerekli olan teorik altyapı incelenip bu işlem bloklarının algoritmaları oluşturulmuştur. Reed–Solomon kodların temeli, matematiksel olarak sonlu alanlara (Galois Field-GF) dayandığı için, kod çözücü tasarımı yapılırken alt bloklardaki tüm işlemler sonlu alanda tanımlı işlemlerdir.

REED-SOLOMON DECODER DESIGN

SUMMARY

Decoders are digital circuits which transforms the input codewords into the coded outputs. Input and output informations have been coded with different ways and the aim of decoders is to obtain the original data. Decoders are commonly used in data multiplexing, 7 segment display and memory address decoding.

In a communication channel, noise symbols are added to the transmitting data. Error Correcting Coding is a process detects the added noises and removes them from the output data. Reed-Solomon(RS) codes are nonbinary BCH codes are a subtype of error correcting codes. This thesis includes informations about RS and BCH codes.

In the content of this thesis RS decoder circuit have been handled considering the RS codes and their properties and implemented using Very High Speed Integrated Circuits Hardware Description Language (VHDL).

In the implementation process, theorical groundwork which are necessary for the design have been examined and created block algortihms. Because of the basic of Reed-Solomon codes are based on finite fields mathematically, all operations in the subblocks are defined in finite fields.

1. GİRİŞ

Haberleşme sistemleri genel olarak kablolu ve kablosuz haberleşme sistemleri olmak üzere ikiye ayrılır. Kablosuz haberleşme sistemleri, temelde, alıcı, kanal ve verici kısımlarından oluşmaktadır. Haberleşme sisteminin içindeki verici bloğu bilgiyi standartlara uygun şekilde işler. Alıcı kısım temel anlamda, verici kısımda iletilmek istenen bilgi işaretini çözmek için kullanılmaktadır. Haberleşme kanalı, bir bilgi işaretini örneğin bir bit dizisini bir ya da daha fazla sayıda alıcıya göndermek için kullanılır.

Haberleşme sistemleri için veri iletiminin etkin ve güvenli bir şekilde gerçekleştirilmesi önemli bir konudur. Fakat bilginin bir iletişim kanalından geçişi ya da kaydedilmesi sırasında veriye istenmeyen fazladan bileşenler eklenebilir. Alıcı kısmındaki kod çözücü, veriye eklenen fazladan bileşenleri iletilen bilginin ilk halini belirlemek için kullanır.

1960 yılında Irving Reed ve Gus Solomon, kendi adlarıyla anılacak olan yeni bir hata düzeltme kodlaması tanımlamışlardır[1]. Bu kodlar, ikili olmayan BCH kodlarının önemli bir alt sınıfıdır[2]. Reed–Solomon kodlar bir hata düzeltme kodlaması (Error Correction Coding-ECC) türüdür ve yıllar önce tanıtılmış olmasına rağmen günümüzde önemli uygulama alanlarına sahiptir.

Günümüzde Reed - Solomon kodların en önemli uygulama alanları:

Bellek elemanları (CD, DVD, etc)
Kablosuz haberleşme (Cep telefonları, mikrodalga bağlantılar)
Uydu haberleşmesi
Geniş bandlı modemler (ADSL, xDSL, etc)

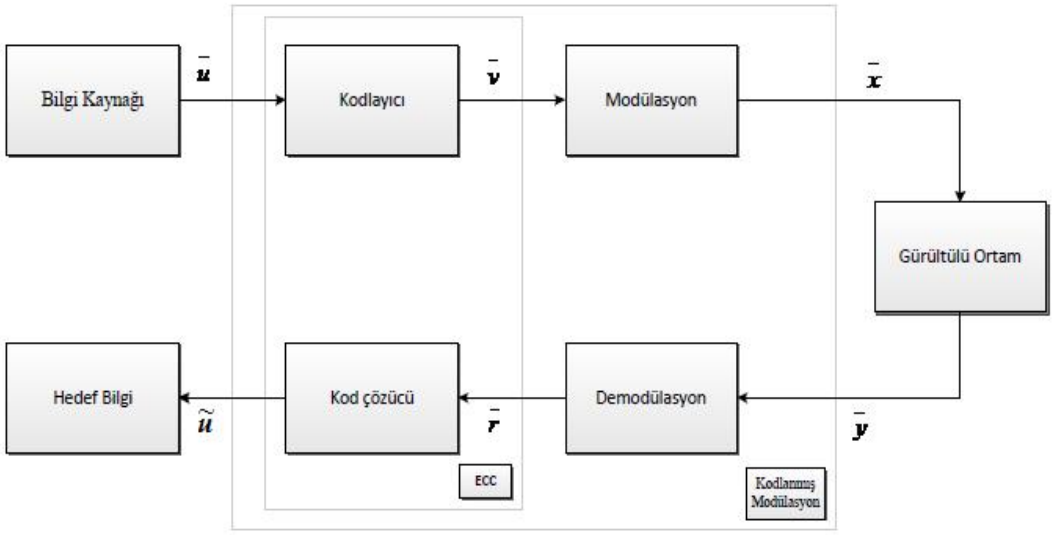
olarak belirtilebilir[2].

Bu bitirme çalışması kapsamında Reed-Solomon kod çözücü devresi ele alınmış ve gerçeklemesi yapılmıştır. Çalışmanın 2. bölümünde RS kodların dahil olduğu hata düzeltme kodlaması hakkında bilgi verilmiş ve lineer blok kodlar açıklanmıştır. İkili olmayan BCH kodların ve dolayısıyla RS kodların matematiksel temeli sonlu alanlara dayandığından sonlu alanlar ile ilgili özellikler 3. bölümde, BCH kodları 4.

bölümde anlatılmıştır. 5. bölümde Reed-Solomon kod çözücünün yapısı hakkında bilgi verilmiş ve RS(255,239) koduna göre VHDL gerçeklemeleri yapılmıştır.

2. HATA DÜZELTME KODLAMASI

Haberleşme sistemlerinin temel amacı, bir ortamdan başka bir ortama geçiş yapan verinin ilk halinin korunmasıdır[3]. Fakat, sayısal bir verinin iletişim kanalından geçişi ya da kaydedilme işlemi sırasında çıkışta elde edilen veri, orijinal verinin fazladan (redundant) bilgi eklenmiş halidir. Şekil 2.1, standart bir iletişim sisteminin blok diyagramını göstermektedir. Bilgi kaynağı ve bilginin ulaşması beklenen hedef, bilginin yapısıyla uyumlu olan herhangi bir kaynak kodlamayı içerir. Hata düzelten kodlayıcı bilgi sembollerini kaynaktan giriş olarak alır ve aldığı sembollere fazladan bilgiler ekler. Bu eklenen fazladan bilgilerin (hataların) çoğu düzeltilebilir.



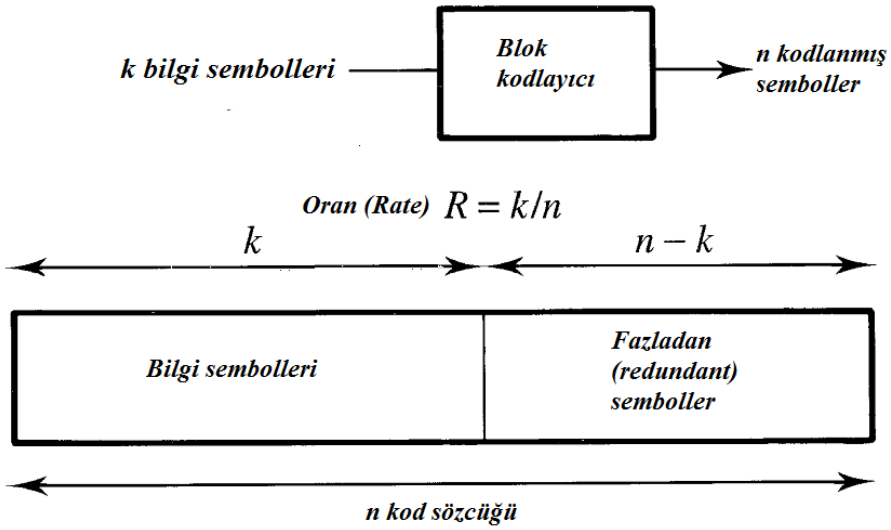
Şekil 2.1: Standart bir iletişim sistemi [2].

Hata düzeltme kodlarının işlevi, çıkışta elde edilen sembol ya da işaretten verinin ilk haline eklenen gürültünün çıkarılmasıdır. Çıkışta elde edilen veri sembolüne kod sözcüğü (codeword) denir [2]. Kanal kodlama, blok kodlama ve katlamalı kodlama olarak iki temel forma ayrılır [3].

2.1 Blok Kodlar

Blok kodlar, bilgiyi bir bloktan diğerine aktararak işler ve her bir blokta ele alınan bilgi diğerlerinden bağımsızdır. Diğer bir deyişle blok kodlama, hafızası olmayan bir işlemdir, bir anlamda kod sözcükleri birbirinden bağımsızdır. Tersine, katlamalı

kodlayıcının çıkışı, sadece o anki girişteki veriye bağlı değildir, aynı zamanda daha önceki giriş ve çıkışlara bağlıdır [2].



Şekil 2.2: Hata düzeltme için sistematik blok kodlama [4].

Blok kodlar, tekrardan iletime gerek kalmadan, sınırlı sayıdaki hatanın algılanmasına ve düzeltilmesine olanak sağlar. Bir blok kod, *k* adet ikili giriş simgesini, *n* adet ikili çıkış simgesine dönüştürür (Şekil 2.2). Yani kod kelimesi içindeki her bit, bir önceki bitten bağımsızdır. $n>k$ olduğundan dolayı, seçilen kod, kod kelimesi uzunluğunda bir artışa sebep olacaktır. Bu artık bitler, eşlik bitlerinde olduğu gibi, kod çözücü tarafından hata algılanmasında ve düzeltilmesinde kullanılacaktır. Blok kodlar (n,k) şeklinde bir gösterime sahiptirler. Bunun yanında kanal kodlayıcıların performansını belirleyen en önemli kavramlardan biri de kod hızı ya da oranı (code rate) olarak tanımlanan $R=k/n$ eşitliğidir. $n>k$ olmasından dolayı kod hızı 1'den küçüktür. Blok kodlar aşağıdaki özelliklere göre sınıflandırılabilir;

Doğrusallık: Bir kodun doğrusal olduğunu söyleyebilmemiz için, kodda ki iki kod kelimesinin toplamının başka bir kod kelimesi oluşturması gerekir. Doğrusal bir kod, tümü sıfır olan bir kod kelimesi içermelidir.

Sistematiklik: Sistematik kod eşlik bitlerinin bilgi bitlerinin sonuna eklendiği bir koddur. Dolayısıyla, gönderilen veri sözcüklerinin kodlayıcıda değişmemesi sonucu kod çözücüde esas koda dönüştürme ihtiyacı olmaz.

Çevrimsellik: Çevrimsel kodlar, blok kodlar ailesinin bir üyesidir. Bir kodun çevrimsel olabilmesi için kod kelimesi bir bit sağa kaydırıldığında ve en sağdan düşen bit sol başa eklendiğinde yeni bir kod kelimesi oluşması gerekir.

Blok kodların kullanımıyla performans başarımında önemli bir artış elde edilmesine rağmen, blok kodların yapısından kaynaklanan birkaç dezavantaj nedeniyle iletişim sistemleri tasarımında kullanılması bazı sorunlara sebep olmaktadır. Bunlar;

Çerçeve uyumluluğu olması nedeniyle kod çözme işlemi başlamadan önce iletilen bütün blokların alıcı tarafından alınması gerekmektedir. Bunun sonucunda, özellikle büyük blok uzunluklarında sistemde kabul edilemeyecek bir gecikme oluşmaktadır. Kesin (çok iyi) bir çerçeve senkronizasyonuna gerek duyarlar.

Kodlama işlemi her zaman için ilgili bilgiye ait kod sözcüğü incelenerek yapılır. Hatalarla karşılaşıldıktan sonra, kanal çıkışında elde edilen bilgi dizisi ile her bir kod sözcüğü karşılaştırılarak alınan bilgiye en yakın kod sözcüğü bulunur. Bu uzaklığa Hamming uzaklığı denir [5].

Blok kodlarda kullanılan kod çözücüler, 0-1kararına (hard decision) dayalı olarak alışmaktadır. 0-1 kararına dayalı olarak çalışan kod çözücülerde, kanal çıkışında alınan bilgi ikili düzende (0 veya 1) olmasına rağmen, yumuşak kararlı (soft decision) kod çözücülerde kanalın çıkışında alınan bilgi sürekli (reel) bir değer olacaktır. Gerçekte, blok kodlarda yumuşak kararlı kod çözücülerin kullanımı mümkün olmasına rağmen, işlem karmaşıklığının artması nedeniyle tercih edilmemektedir [3].

Blok kodların dezavantajlarını gidermek için, farklı bir yaklaşım olan ve ilk kez 1955'te Elias tarafından ileri sürülen katlamalı (convolutional) kodlar önerilmiştir [4].

3. SONLU ALANLAR (Galois Field - GF)

Sonlu alanlar, Fransız matematikçi Evariste Galois'in (1811-1832) bu alanda yaptığı çalışmalar sonrasında onun adıyla anılmaya başlamıştır. q sayıda eleman bulunduran bir Galois Alan GF(q) ile gösterilir ve bu şekildeki en basit alan GF (2) dir [7].

Reed–Solomon gibi ikili olmayan kodlarda kodlama ve kod çözme süreçleri düşünüldüğünde Galois Field (GF) olarak bilinen sonlu alanları ve özelliklerini anlamak gereklidir.

Tamsayılar kümesi gibi bazı alanlar sonsuzdur ve bu alanların elemanları bilgisayarda sabit uzunluktaki sözcüklerle gösterilemeye çalışıldığında bazı sorunlar ortaya çıkar.

GF alanda tanımlı herhangi bir elemanın üzerinde yapılan herhangi bir işlem yine GF alana tanımlı bir elemana denk düşer. GF alanda tanımlı elemanlar sabit uzunluklu bir ikili sözcük ile gösterilebildiğinden bunlara Sonlu Alanlar (Finite Fields) denir .

p sonlu alanda bir asal sayı olduğunda sonlu alandaki elemanların sayısı bu asal sayının kuvveti olmalıdır ($q=p^r$) . Her bir asal sayı üssü için ($q=p^r$) q dereceli bir alan vardır ve bu alan tektir. F_0 alanı; p ye göre modu alınmış tamsayıları içeriyorsa bu alanda r dereceli bir indirgenemez (irreducible) $f(X)$ polinomu bulunur ve bu polinomun baş katsayıları 1 e eşittir.

Geleneksel matematiksel yaklaşımda alanın içindeki herhangi bir aritmetik işlemin alanın dışındaki bir elemanla sonuçlandıktan sonra alan içindeki bir elemanla eşleştirilmesi işlemine mod alma işlemi denir. Sonlu alanda elemanların kolay kullanılabilmesi için elamanlar ikili eşdeğerleri ile ifade edilirler.

Galois alan, idealde donanım uygulamalarına uygundur. Toplama ve çıkarma işlemleri iki ya da daha fazla elemanın XOR işlemine girmesi anlamına gelmektedir. Çarpma işlemi daha karmaşıktır, bazı kombinezonsal lojik işlemler ve kaydediciler kullanılarak gerçekleştirilir. Tezin sonraki kısmında çarpma işlemi ve gerçeklemesi ile ilgili detaylı bilgi verilecektir.

3.1 Sonlu Alanların Yapısı

p asal sayısı için p elemanlı GF(p) ile gösterilen bir sonlu alan vardır bu alanı p^m elemanlı bir alana genişletmek mümkündür. p^m elemanlı alana GF(p) nin genişletme alanı (extension field) denir ve m pozitif bir katsayıya eşit iken GF(p^m) ile gösterilir. Buna göre, GF(p) nin elemanları GF(p^m) nin elemanlarının alt kümesidir. Reed – Solomon kodlarının yapısında GF(2^m) genişletme alanındaki semboller kullanılırlar [8].

Reel sayılar alanının, karmaşık sayılar alanının alt alanı olmasına benzer şekilde ikili sonlu alan GF (2) de GF(2^m) genişletme alanının alt kümesidir. Genişletme alanında 0 ve 1 sayılarının yanında ek olarak α ile gösterilen bir sembol vardır. GF(2^m) deki sıfır olmayan her bir eleman α nın kuvveti olarak gösterilebilir. Elemanların oluşturduğu sonsuz F kümesi {0, 1, α} elemanlarından başlayarak oluşturulur ve diğer elemanlar; bir önceki elemanın α ile çarpılmasıyla üretilir.

$$F=\{0, 1, \alpha, \alpha^2, \ldots,, \alpha^j, \ldots.\} = \{0, \alpha^0, \alpha^1, \alpha^2, \ldots,, \alpha^j, \ldots.\}[8] \qquad (3.1)$$

F kümesi, gerçekte 2^m tane eleman içerir ve çarpma işlemine göre kapalıdır. $p(X)$ GF(2) nin m dereceli asal polinomu ise sonlu alanda 1 ve 0 sayılarına ek olarak tanımlı diğer eleman olan α elemanı p(X) asal polinomunun köküdür [8]. GF(2^m) in asal elemanı α, $n = 2^m - 1$ için,

$$\alpha^n=1 \qquad (3.2)$$

eşitliğini sağlar [2].

Örnek 3.1 : GF(2^3) alanının asal polinomu $p(X) = x^3 + x + 1$ ve α, $p(X)$ asal polinomunun kökü ise $p(\alpha) = \alpha^3 + \alpha + 1 = 0$ olur. Buna göre, Tablo 2.1 GF(2^3) te bulunan elemanlarının 3 değişik gösterimini açıklamaktadır.

Tablo 2.1: GF(2^3) elemanlarının 3 farklı gösterimi [2].

Kuvvet	Polinom	Vektör
-	0	000
1	1	001
α	α	010
α^2	α^2	100
α^3	1+ α	011
α^4	$\alpha + \alpha^2$	110
α^5	$1+ \alpha + \alpha^2$	111
α^6	$1 + \alpha^2$	101

$\alpha^3 + \alpha + 1 = 0$ ise $\alpha^3 = \alpha + 1$

$$\alpha^4 = \alpha \,.\, \alpha^3 = \alpha \,.\, (1+ \alpha) = \alpha + \alpha^2$$

$$\alpha^5 = \alpha \,.\, \alpha^5 = \alpha \,.\, (\alpha + \alpha^2) = \alpha^2 + \alpha^3 = 1 + \alpha + \alpha^2$$

$$\alpha^6 = \alpha \,.\, \alpha^3 = \alpha \,.\, (1+ \alpha + \alpha^2) = \alpha + \alpha^2 + \alpha^3 = \alpha + \alpha^2 + 1 + \alpha = 1 + \alpha^2$$

3.2 Sonlu Alan Aritmetiği

GF(2) sonlu alanı, elemanları 0 ve 1 olan -ikili sayılardan oluşan- bir sayı sistemidir ve toplama ve çarpma işlemleri aşağıdaki kurallara uyar:

0+0=1+1=0

0+1=1+0=1

0x0=1x0=0x1=0

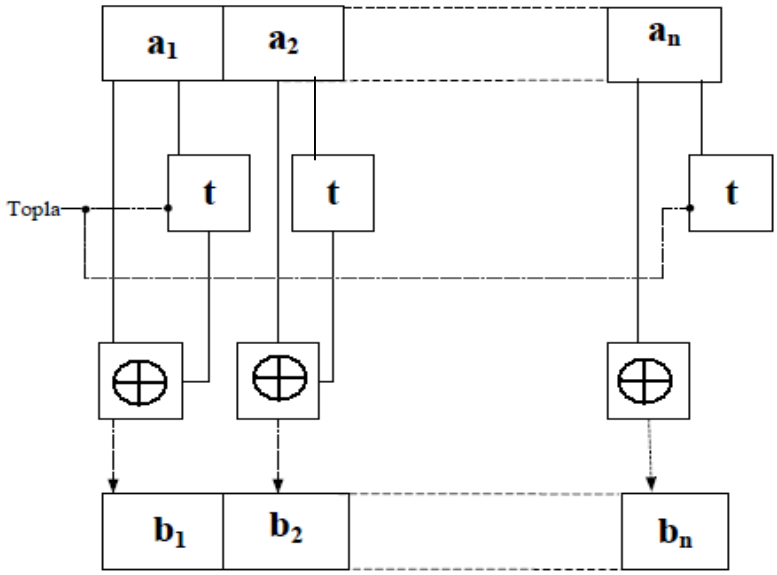
1x1=1

Bu kurallar mod 2 aritmetiği olarak bilinir.

3.2.1 Sonlu Alanda Toplama İşlemi

Şekil 2.3 te görüldüğü üzere, *n* sayıda sıralı ikiliyi toplayan kombinezonsal yapı *n* sayıda Boolean denkleminin $(a_1+b_1,a_2+b_2,\ldots..,a_n+b_n)$ uygulamasıdır., $\oplus$ sembolü *mod2* ye göre toplama işlemini belirtir.GF (2^n) de *n* bitlik 2 sayının toplanması (ya da çıkarılması) anlamına gelmektedir.

GF(2^m) de çarpma işlemini gerçekleştiren kombinezonsal yapının 2*m* sayıda girişi, *m* sayıda çıkışı vardır. Dolayısıyla, *m*_sayıda Boolean denklemi türetilmelidir. GF (2^m) alanındaki 2^m eleman için toplama ve çarpma kuralları o alana ait olan *m* dereceli indirgenemez polinom ile ilişkilidir. İşlemler indirgenemez polinoma göre mod alınarak yapılır.



Şekil 2.3: Galois Alanda Toplayıcı [9].

Sonlu alanda toplama ve çıkarma işlemleri basit bir XOR işlemi olduğundan bu işlemlerde vektör gösterimi en kullanışlı olandır. Fakat elemanlar çarpılırken elemanların Tablo 2.1 in ilk sütununda belirtilen asal elemanın kuvvetleri olarak gösterilmesi daha uygundur.

3.2.2 Sonlu Alanda Çarpma İşlemi

A(x) ve B(x) polinomları çarpılacak sayıları temsil ederken, P(x) polinomu da ilgili GF(2^m) alanındaki indirgenemez polinomu temsil ettiğinde, çarpma işlemi aşağıdaki gibi tanımlanır:

$$C(x)=A(x)xB(x)\text{mod}P(x) \qquad (3.3)$$

Bir polinomun matematiksel ifadesi;

$$A(x)=\sum_{i=0}^{m-1} a_i x^i = a_{m-1}x^{m-1}+a_{m-2}x^{m-2}+\ldots+a_1x+a_0 \qquad (3.4)$$

a_i değerleri GF(2) nin elemanlarıdır ve A sayısındaki 0 ve 1 lerin yerine denk düşerler m bu simgedeki bit sayısını ifade eder.

Örnek 3.2: A ve B; GF(2^8) de tanımlı ve değerleri sırasıyla "10001001" ve "01011001" olan sayılar olsun.

2.6 eşitliğinden hareketle bu sayıların polinom olarak gösterilimi aşağıdaki gibidir :

$A(x) = x^7 + x^3 + 1$, $B(x) = x^6 + x^4 + x^3 + 1$

A(x) x B(x) : $(x^7 + x^3 + 1).(x^6 + x^4 + x^3 + 1) = x^{13} + x^{11} + x^{10} + x^7 + x^9 + x^7 + x^6 + x^3 + x^6 + x^4 + x^3 + 1$

A(x) x B(x) işlemi basit bir XOR toplamaya dönüşür. XOR toplamada 1 + 1 = 0 ve 0 + 0 = 0 olduğundan toplama işlemine giren polinomların aynı dereceli terimlerinin başında aynı katsayı varsa bu terimler çıkışta görülmez. Dolayısıyla;

A(x) x B(x) = $x^{13} + x^{11} + x^{10} + x^9 + x^4 + x^0$ bulunur.

GF(2^8) deki asal polinom, P(x) = $x^8 + x^4 + x^3 + x^2 + 1$ olduğundan çarpma işleminin sonucu;

C(x) = $x^{13} + x^{11} + x^{10} + x^9 + x^4 + 1$ mod $(x^8 + x^4 + x^3 + x^2 + 1)$

$$\begin{array}{l|l}
x^{13}+x^{11}+x^{10}+x^9+x^4+1 & x^8+x^4+x^3+x^2+1 \\
\underline{-\,x^{13}+x^9+x^8+x^7+x^5} & \\
x^{11}+x^{10}+x^8+x^7+x^5+x^4+1 & x^5+x^3+x^2+1 \\
\underline{-\,x^{11}+x^7+x^6+x^5+x^3} & \\
x^{10}+x^8+x^6+x^4+x^3+1 & \\
\underline{-\,x^{10}+x^6+x^5+x^4+x^2} & \\
x^8+x^5+x^3+x^2+1 & \\
\underline{-\,x^8+x^4+x^3+x^2+1} & \\
x^5+x^4 &
\end{array}$$

Reel alanda olduğu gibi sonlu alanda da mod alma işlemi, alan içinde olmayan bir sayının sonuçta alan içinde bir eleman elde edinceye kadar modu alınacak sayıya bölünmesiyle yapılır.

Çarpımdan elde edilen C sayısının polinom gösterimi $C(x)=x^5+x^4$ olduğuna göre; C sayısı "00110000" bulunur.

3.3. Sonlu Alanda Eşlenik ve Minimal Polinom Kavramları

f(X); GF(2) de tanımlı bir polinom ve ß, GF(2^m) de tanımlı bir eleman ve *f(X)* polinomunun kökü olsun. Herhangi bir $l \geq 0$ tamsayısı için $ß^{2^l}$ de bu denklemin köküdür. Matemetiksel olarak;

$$[f(ß)]^{2^l}=(0)^{2^l}=f(ß^{2^l})=0 \qquad (3.5)$$

$ß^{2^l}$ elemanına ß nın eşleniği denir [8]. Sonuç olarak, ß elemanı genişletilmiş alan GF(2^m) de bulunuyor ve *f(X)* polinomunun kökü ise eşlenikleri de aynı alanın elemanlarıdır ve aynı polinomun kökleridir.

GF(2^m) üzerinde tanımlı, yine aynı alanda tanımlı ß değerini kök olarak kabul eden ve en küçük dereceli Ø (*X*) polinomuna ß nın minimal polinomu denir. Buna göre, 0 elemanının minimal polinomu *X*, 1 elemanının minimal polinomu $1 + X$ tir [10].

Ø (*X*), GF(2^m) deki bir ß elemanının minimal polinomu ve *e;* $ß^{2^e}$ = ß eşitliğini sağlayan en küçük tamsayı ise [8]; ;

$$\phi(X) = \text{[illegible]}=0\text{[illegible]}-1\text{[illegible]}+ß^{2^l} \qquad (3.6)$$

Örnek 3.2: GF(2^3) alanında tanımlı $ß = \alpha^3$ ise ß nın eşlenikleri:

$ß^2 = \alpha^6$, $ß^{2^2} = \alpha^6$, $ß^{2^3} = \alpha^{24} = \alpha^9$

$ß = \alpha^3$ ün minimal polinomu:

$Ø(X) = (X + \alpha^3)(X + \alpha^6)(X + \alpha^{12})(X + \alpha^9)$ olarak bulunur.

Eşitliğin sağ tarafı düzenlenirse;

$$Ø(X) = [X^2 + (\alpha^3 + \alpha^6)X + \alpha^9][X^2 + (\alpha^{12} + \alpha^9)X + \alpha^{21}]$$

$$= (X^2 + \alpha^2 X + \alpha^9)(X^2 + \alpha^8 X + \alpha^6)$$

$$= X^4 + (\alpha^2 + \alpha^8)X^3 + (\alpha^6 + \alpha^{10} + \alpha^9)X^2 + (\alpha^{17} + \alpha^8)X + \alpha^{15}$$

$= X^4 + X^3 + X^2 + X + 1$ bulunur.

4. BCH KODLARI

Bose-Chaudhuri-Hocquenghem kodları, kodlama ve kod çözme işlemlerini kolaylaştırmada kullanılan çevrimli kodların bir türüdür [2]. İkili kodların Bose ve Ray-Chaudhuri ve onlardan bağımsız olarak Hocquenghem tarafından keşfedilmesi, cebirsel yapıya dayalı kodların araştırılmasında olağanüstü bir başarıdır [11]. İkili BCH kodlar, 1961 yılında Gorenstein ve Zierler tarafından p asal olmak üzere p^m sembole genişletilmiştir [12]. İkili olmayan BCH kodların arasında en önemli alt sınıf Reed–Solomon (RS) kodlardır.

4.1. İkili BCH Kodları

$m(m\geq 3)$ve $t(t<2^{m-1})$ herhangi iki pozitif tamsayı olmak üzere, aşağıdaki parametrelere sahip bir BCH kodu bulunur [6]:

Blok uzunluğu: $n=2^m-1$, (4.1.a)

Fazladan sembollerin sayısı: $n-k\leq mt$, (4.1.b)

Minimum uzaklık: $d_{min}\geq 2t+1$ (4.1.c)

Yukarıda özellikleri verilen (n,k) kodu $n = 2^m - 1$ kod sözcüğündeki t ya da daha az sayıdan oluşan hataları düzeltebilir. Bu kodun üretici polinomu, Galois alanın GF(2^m) kökleri tarafından belirlenir. $2^m - 1$ uzunluklu BCH kodun üretici polinomu **g** (X) in kökleri:

$$\alpha, \alpha^2, \alpha^3, \ldots, \alpha^{2t}$$

ise;

$1\leq i \leq 2t$ için $\mathbf{g}(\alpha^i)=0$ olmalıdır.

$\boldsymbol{\emptyset}_i(X)$, α^i nin minimal polinomu olsun. İkili bir BCH (n, k, d_{min}) BCH kodun üretici polinomu:

$$\boldsymbol{g}(X)=\mathrm{LCM}\{\boldsymbol{\emptyset}_1(X), \boldsymbol{\emptyset}_2(X), \ldots . \boldsymbol{\emptyset}_{2t-1}(X)\} \quad (4.2)$$

2.3 eşitliğinden görüldüğü görüldüğü üzere **g**(X) üretici polinomu, **g**(X) in her bir kökü için var olan minimal polinomların En Küçük Ortak Çarpanı (Least Common Multiple–LCM) na eşittir [6].

Örnek 4.1: GF (2^3) alanında tanımlı $p(x) = x^4 + x + 1$ polinomu için üretici polinom (t =3):

g (X) = LCM{$\boldsymbol{\emptyset}_1$(X), $\boldsymbol{\emptyset}_2$(X)}= $x^3 + x + 1$ olarak bulunur.

4.2. İkili Olmayan BCH Kodları

4.2.1. Reed – Solomon (RS) Kodlar

1960 yılında, Irving Reed ve Gus Solomon tarafından tanımlanan yeni bir hata düzeltme kodlama sınıfı olan kodlar daha sonra Reed – Solomon kodlar olarak onların adıyla anıldı [1].1964 te Singleton, aynı uzunluk ve boyuttaki diğer kodlarla karşılaştırıldığında RS kodların en iyi hata düzeltme kapasitesine sahip olduğunu gösterdi [13].

Reed Solomon kodlar, m 2 den büyük herhangi bir tamsayı iken, m bitlik serilerden oluşan semboller ile ifade edilen ikili olmayan çevrimsel kodlardır.

t sayıda hata düzeltebilen bir RS kodu, aşağıdaki gibi gösterilebilir [14] :

$$(n,k,t)=(2^m - 1,\ 2^m - 1 - 2t,\ t) \tag{4.3}$$

4.3 gösteriminde k, kodlanan veri sembollerinin sayısını, n ise kodlanmış bloktaki kod sembollerinin toplam sayısını, t sembolü ise kodun hata düzeltebilme kapasitesini göstermektedir [8].

GF(q) alanının asal elemanı α olduğunda t sayıda hata düzeltebilen RS kodun üretici polinomuun tüm kökleri α, α^2, . . . , α^{2t} olur. α^i değerleri, GF (q) nun elemanı olduğundan bu elemanın minimal polinomu basitçe X - α^i şeklindedir. Buna göre, 4.2 eşitliğinden hareketle, üretici polinom [6]:

$$\mathbf{g}(X)=(X-\alpha)(X-\alpha^2)\ \ldots(X-\alpha^{2t}) \tag{4.4}$$

$$= g_0 + g_1 X + g_2 X^2 + \ldots + g_{2t-1} X^{2t-1} + X^{2t}$$

bulunur.

5. REED–SOLOMON KOD ÇÖZÜCÜ

Bölüm 2 de belirtildiği üzere, bir haberleşme sistemine giren veri sembolüne verinin sistemden geçişi sırasında bazı fazlalık bilgiler (hatalar) eklenir. Sistemin çıkışından alınan ilk verinin hata sembolleri eklenmiş haline kod sözcüğü denir.

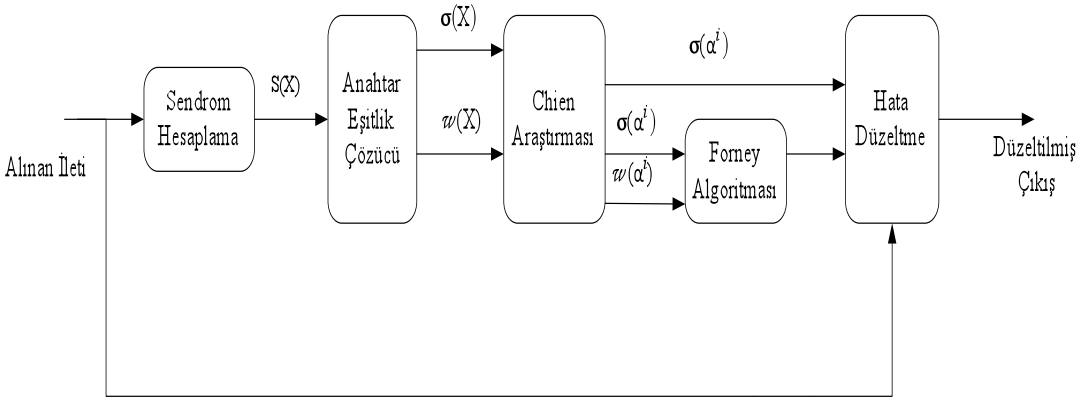
Bir haberleşme kanalından iletilen veriye ait kod polinomu $\mathbf{V}(X) = V_0 + V_1 X + \ldots + V_{n-1} X^{n-1}$ ve bu iletilen veri, haberleşme kanalından geçtikten sonra çıkışta alınan polinom $\mathbf{R}(X) = R_0 + R_1 X + \ldots + R_{n-1} X^{n-1}$ olsun.

Bu durumda kanal tarafından eklenen hata sembolü;

$$\mathbf{E}(X) = \mathbf{R}(X) - \mathbf{V}(X) \text{ olur.} \tag{5.1}$$

Reed–Solomon kod çözme, kanal çıkışında elde edilen ***R*** verisini kullanarak kanal girişindeki verinin orijinal hali **V** nin hesaplanması işlemidir.

Şrkil 5.1 den görüldüğü üzere, bir RS(n,k) kodu için kod çözme işleminde ilk aşama, $0 \leq i \leq 2t-1$ olmak üzere S_i ile gösterilen $2t$ sayıdaki sendrom değerlerini bulmaktır. İkinci blok olan Anahtar Eşitlik Çözücü (Key Equation Solver - KES) bloğunda hata yerleri ve değerlerini temsil eden $\sigma(x)$ ve $\omega(x)$ polinomları hesaplanır. Forney ve Chien Araştırması devreleri bu polinomların katsayılarını hesaplar ve bu işlemlerin sonucunda herhangi bir iletişim kanalında iletilen veriye eklenen hataların yerleri ve değerleri tespit edilmiş olur.



Şekil 5.1: Reed – Solomon kod çözücü blok diyagramı[15].

Bu bitirme çalışmasında 8 hatayı düzeltebilen RS(255,239) kodu ele alınmış ve VHDL gerçeklemeleri bu koda göre yapılmıştır.

5.1. Sendrom Hesaplama Bloğu

RS (n, k) kodu için hesaplanan S_i değerlerini temsil eden S(x) polinomu

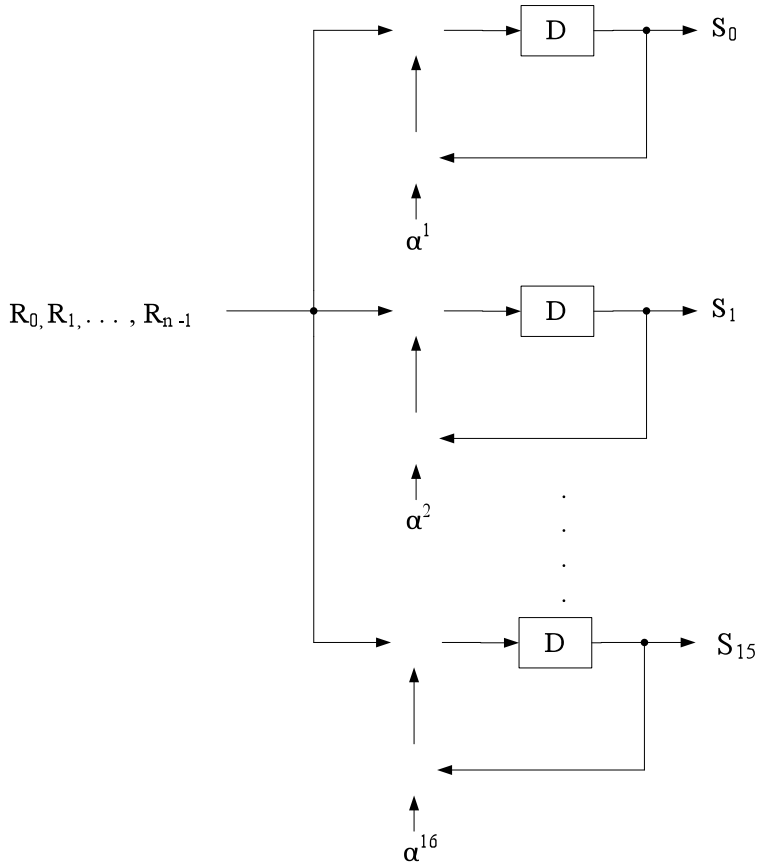
$S_i = \sum_{j=0}^{2t-1} r_j \cdot \alpha^{ij}$ olacak şekilde

(5.2.a)

$S(x) = \sum_{i=0}^{2t-1} S_i \cdot x^i$ şeklindedir.

(5.2.b)

RS(255,239) kodu için $2^m-1=255$ olduğundan $m=8$ bulunur ve matematiksel işlemler $GF(2^m)=GF(2^8)$ alanında tanımlanmalıdır. Bu alanın indirgenemez polinomu $p(x) = x^8+x^4+x^3+x^2+1$ ve 5.1.a eşitliğindeki α^i değeri bu polinomun köküdür, dolayısıyla $GF(2^8)$ alanının asal elemanıdır [16].



Şekil 5.2: Sendrom hesaplama bloğu [16].

Şekil 5.2 yardımıyla S_i değerleri için;

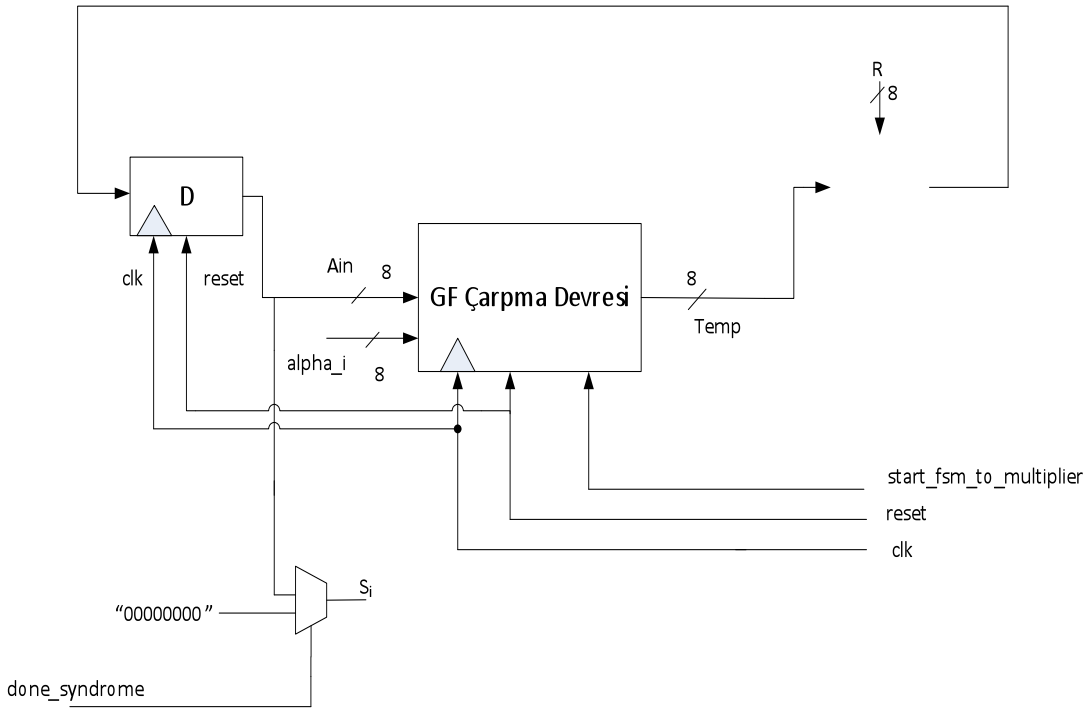
$$S_i=R_0+\alpha^{i+1}.(R_1+\alpha^{i+1}.(R_2+. . . +\alpha^{i+1}(R_{n-1})) \quad (5.3)$$

eşitliği yazılabilir.

Şekil 5.2 de görüldüğü üzere sendrom hesaplama devresinde her bir sendromun hesaplanabilmesi için birer tane toplama bloğu, çarpma bloğu ve kaydedici gerçeklenmelidir. Blok girişinde görülen R_i ($0 \leq i \leq 255$) değerleri, 255 tane 8 bitlik kod sözcüğünden birini temsil eder. Gerçekleme aşamasında her bir sendrom hücresi için kullanılan α değeri aşağıdaki gibidir:

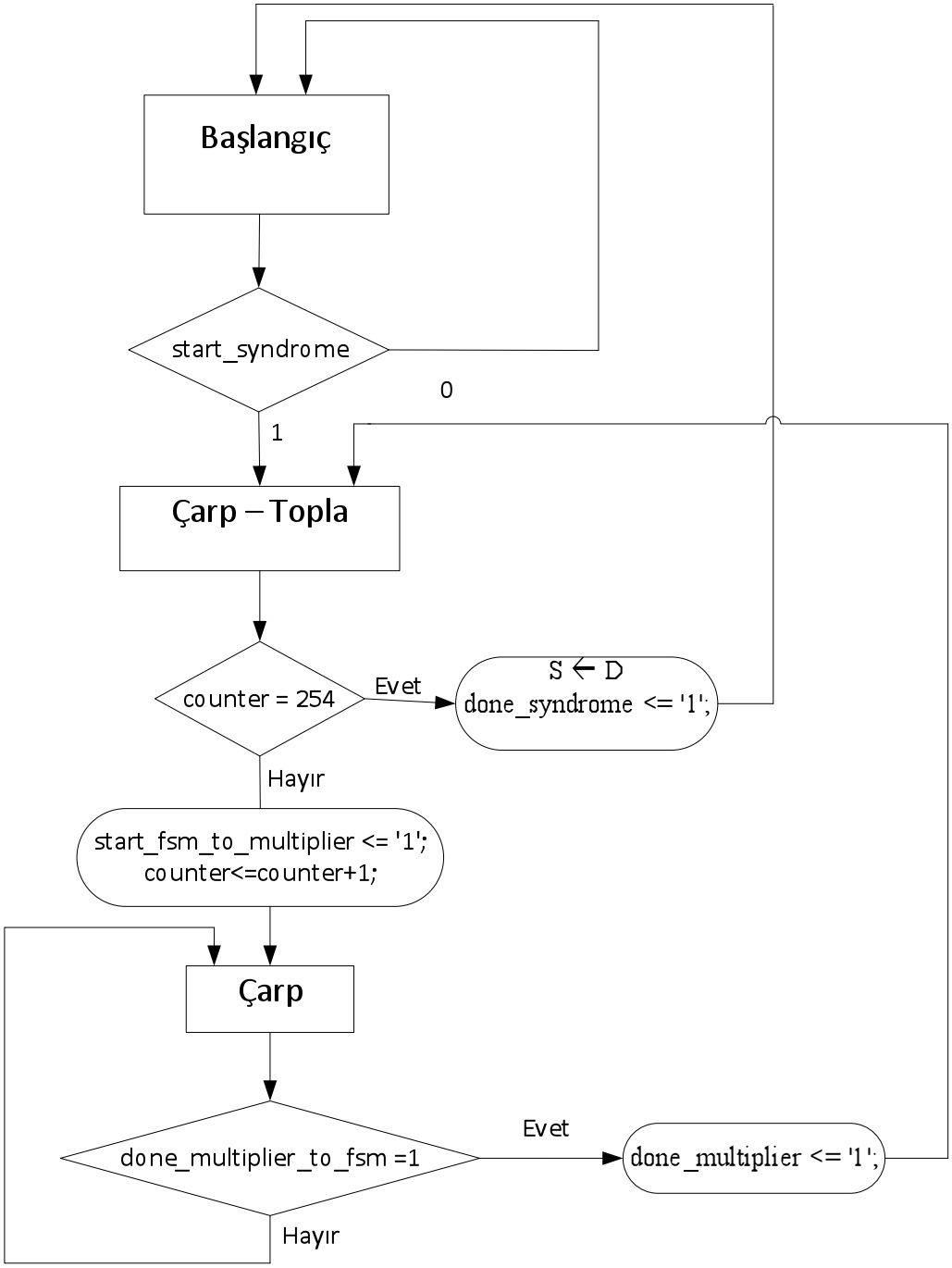
Tablo 5.1: Sendrom Hesaplama Bloğu Gerçeklemesinde Kullanılan α Değerleri

α^1	00000010	α^9	00111010
α^2	00000100	α^{10}	01110100
α^3	00001000	α^{11}	11101000
α^4	00010000	α^{12}	11001101
α^5	00100000	α^{13}	10000111
α^6	01000000	α^{14}	00010011
α^7	10000000	α^{15}	00100110
α^8	00011101	α^{16}	00100110

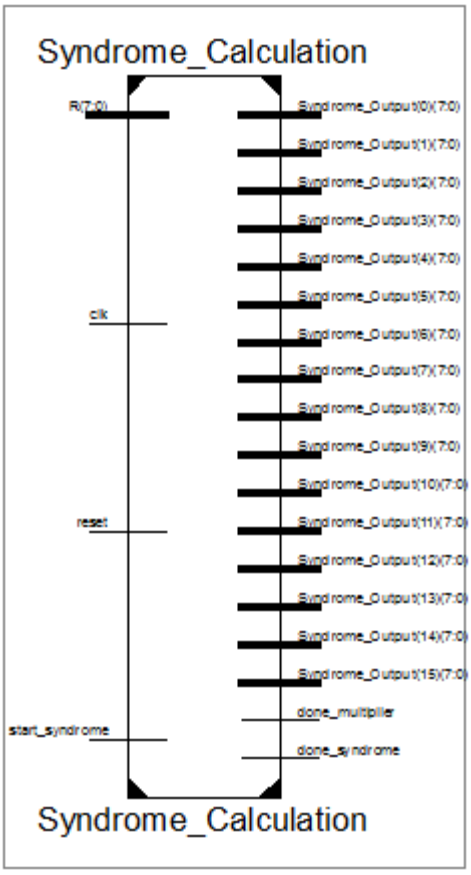


Şekil 5.3: Gerçeklenen Bir Sendrom Hücresinin Blok Diyagramı

Şekil 5.3 te tek bir α değeri için tek bir sendrom değerini hesaplayabilen blok diyagram verilmiştir. Buna göre, XOR bloğunun girişindeki 255 tane R değeri için çarpma ve toplama işlemleri tekrar edilir. Hücrenin girişine R değerlerinin bittiği bilgisi geldiğinde (done_syndrome=1 olduğunda) D kaydedicisinin çıkışındaki değer hücrenin çıkışına (S_1) atanacaktır. Diğer bütün durumlarda S_i "00000000" olacaktır. Şekil 5.4 te görüldüğü üzere, R değerlerini teker teker devreye giriş olarak almak için tüm sendrom hesaplama devresini kontrol eden bir sonlu durum makinası (Finite State Machine-FSM) tasarlanmıştır.

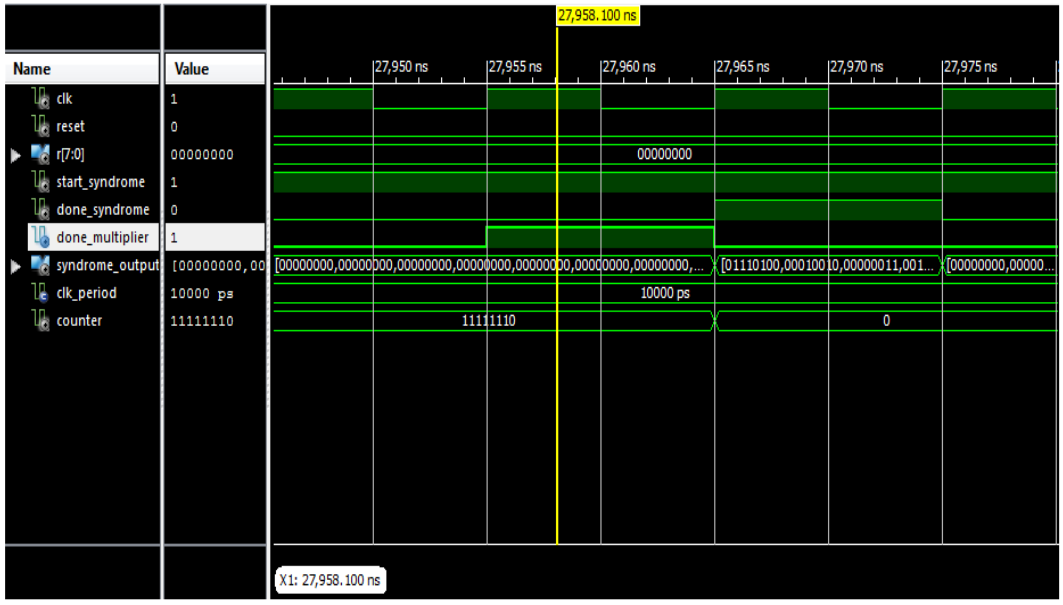


Şekil 5.4: Sendrom Hesaplama İçin FSM



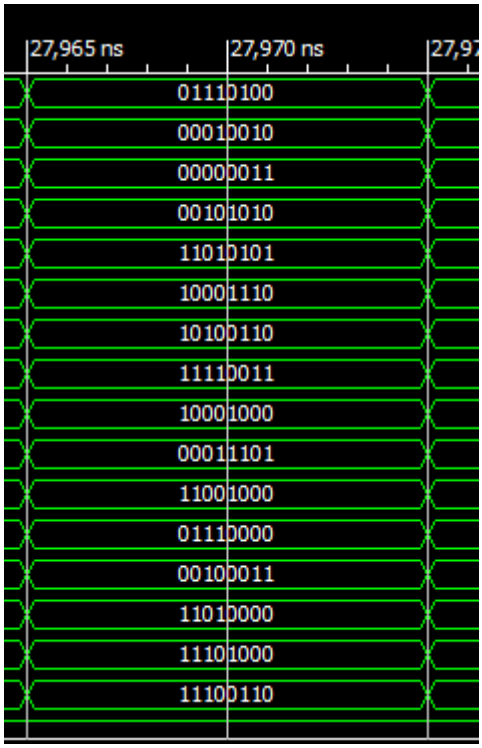
Şekil 5.5: Sendrom Hesaplama RTL Şeması

Şekil 5.5 te çıkışta 16 tane 8 bitlik sendromun alındığı gerçeklenen sendrom hesaplama devresinin RTL şeması verilmiştir.



Şekil 5.6.a: Sendrom Hesaplama Devresi Simülasyon Sonucu

Şekil 5.6.a da görüldüğü üzere sayıcı (counter) değişkeni 254 olduktan sonraki saat darbesinde (255 tane R değeri için işlem yapıldıktan sonra) sendrom değerleri çıkışta görülür.



Şekil 5.6.b: Hesaplanan Sendrom Değerleri

Şekil 5.6.b de R_o = 00001010, R_1 = 00000000, R_2 = 00000000, R_3 = 00000000,

R_4 = 00001010, R_5 = 00000000 ve R_6 = 00001010 ve $7 \leq i \leq 254$ olmak üzere

R_i = 00000000 girişleri için hesaplanan 16 tane 8 bitlik sendrom değeri gösterilmiştir.

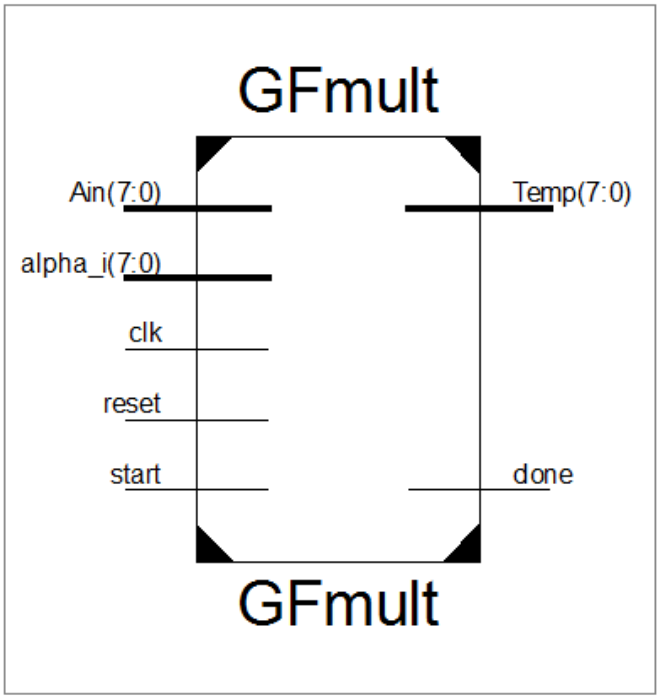
Hesaplanan 16 tane sendrom değerlerinin 0 olması durumunda iletişim kanalına giren $\boldsymbol{V}(x)$ giriş polinomu ve çıkıştan alınan $\boldsymbol{R}(x)$ kod sözcüğü birbirine eşittir, diğer bir deyişle iletişim kanalında veriye eklenen hata polinomu $\boldsymbol{E}(x)$, 0 dır. $\boldsymbol{E}(x) > 0$ ise kod çözücü, hataların değerlerini ve yerlerini belirlemek için;

$$S(x)=S_0+S_1\,x+\ldots+S_{15}\,x^{15}$$

olarak tanımlanan sendrom polinomunu kullanır [17].

5.1.1. Çarpma Bloğu

Tablo 5.1 de görüldüğü üzere α değerleri de R kod sözcükleri gibi 8 bitlik sayılardır. Bir sendrom hücresindeki karmaşık yapı çarpma bloğudur. Eşitlik 3.5 ten de görüldüğü üzere sonlu alanda çarpma işlemi yapılırken çarpılacak sayıları temsil eden polinomlar çarpılıp ilgili alandaki indirgenemez polinoma göre mod alınmalıdır.



Şekil 5.3: Çarpma Bloğu RTL Gösterimi

Şekil 5.3 te 2 tane 8 bitlik sayıyı çarpan sayısal devrenin RTL şeması gösterilmiştir. Şekilde Ain ve alpha_i girişleri 8 bitlik 2 sayıyı, Temp çıkışı da 8 bitlik çarpma işleminin sonucunu göstermektedir. Ain ve alpha_i sayıları polinom olarak gösterimi:

$A(x) = \sum_{i=0}^{7} a_i x^i = (a_7 x^7 + a_6 x^6 + \ldots + a_0 x^0)$

$alpha\_i(x) = \sum_{i=0}^{7} \alpha\_i_i x^i = (\alpha\_i_7 x^7 + \alpha\_i_6 x^6 + \ldots + \alpha\_i_0 x^0)$

Çarpma işleminde öncelikle alpha_i nin en yüksek anlamlı biti, α_i_7, A(x) polinomu ile çarpılır sonuç R(x) kaydedicisine yazılır. İlk basamakla birlikte çarpma işleminin diğer basamakları aşağıdaki adımlarla özetlenmiştir:

1) $R(x)=A(x).\alpha\_i_7$

2. adımda ilk adımda elde edilen R(x) polinomu 1 basamak sola ötelenir ve sonuna 0 eklenir, ötelenmiş R(x) kaydedicisine A(x) polinomu ile alpha_i nin sıradaki bitinin çarpımı eklenir. Bu adımın sonunda xR(x) 9 bitlik bir sayı olabileceği için yenilenen R(x) polinomu da 9 bitlik bir sayı olabilir.

2) $R(x)=x.R(x)+A(x).\alpha\_i_6$

$R(x) = \sum_{i=0}^{8} r_i x^i = (r_8 x^8 + r_7 x^7 + \ldots + r_0 x^0)$

R sayısının 9 bitlik olması durumunda çarpma işleminin 3. adımında R(x) in p(x) indirgenemez polinomuna göre indirgeme yapılmalıdır. 2. adımdaki R(x) polinomunun 9 bitlik bir sayıyı temsil etmesi, en yüksek anlamlı bitinin 1 olması anlamına gelmektedir. Dolayısıyla $r_8=1$ ise mod alma işlemi yapılarak R(x) polinomu yenilenir.

3) $R(x) = R(x)+r_8.p(x)$

Çarpma işleminin geri kalan adımları benzer şekilde devam eder:

4) $R(x)=x.R(x)+A(x).\alpha\_i_5$

5) $R(x)=R(x)+ r_8.p(x)$

•

•

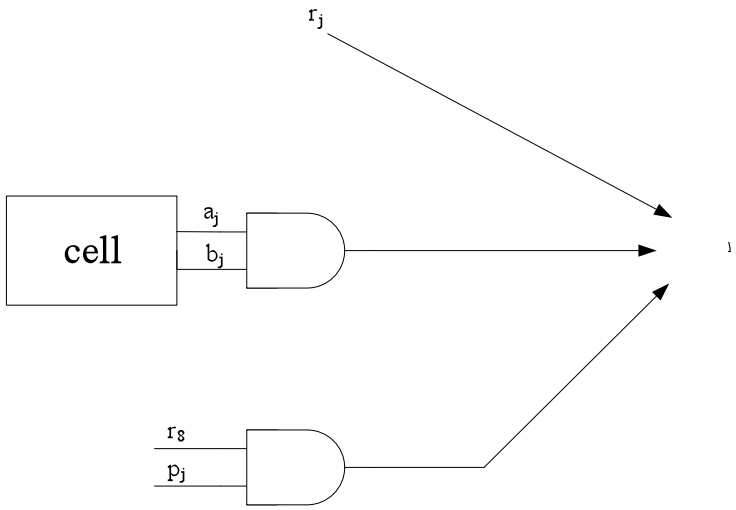
•

Yukarıda sözü edilen adımlar dikkate alındığında çarpma işlemi aşağıdaki durumu gelir:

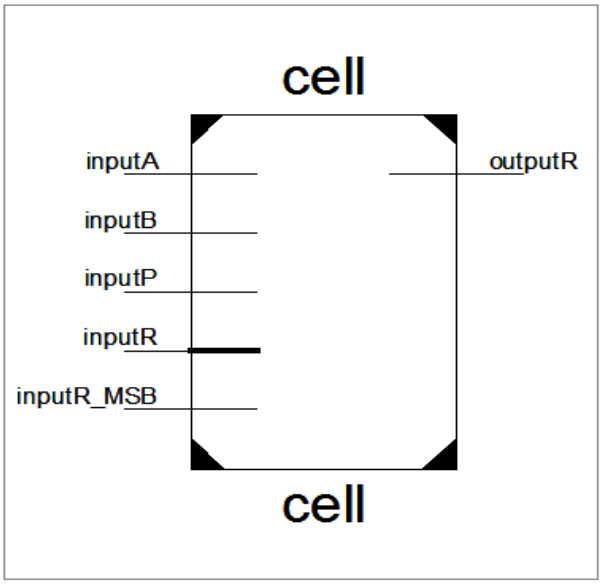
r_8	r_7	r_6	r_5	r_4	r_3	r_2	r_1	0
	$a_7.\alpha\_i_7$	$a_6.\alpha\_i_6$	$a_5.\alpha\_i_7$	$a_4.\alpha\_i_4$	$a_3.\alpha\_i_3$	$a_2.\alpha\_i_2$	$a_1.\alpha\_i_1$	$a_0.\alpha\_i_0$
r_8p_8	r_8p_7	r_8p_6	r_8p_5	r_8p_4	r_3p_3	r_2p_2	r_1p_1	r_0p_0
+								

Her bir basamaktaki toplama işlemini bir hücre olarak kabul ederek çarpma bloğunun tasarımını bu hücrelerin birleştirilmesi işlemine dayandırabiliriz. Birim hüzrenin

yapısı $j = 0,1, \ldots ,7$ olmak üzere şekil 5.4.a daki gibidir. İşlemin sonucunda elde edilen sayı 7 bitlik olmak zorunda olduğundan r_8 ve r_8p_8 bitlerinin toplandığı hücrenin sonucu 0 olmak zorundadır. Dolayısıyla bu hücrenin gerçeklenmesine gerek yoktur.



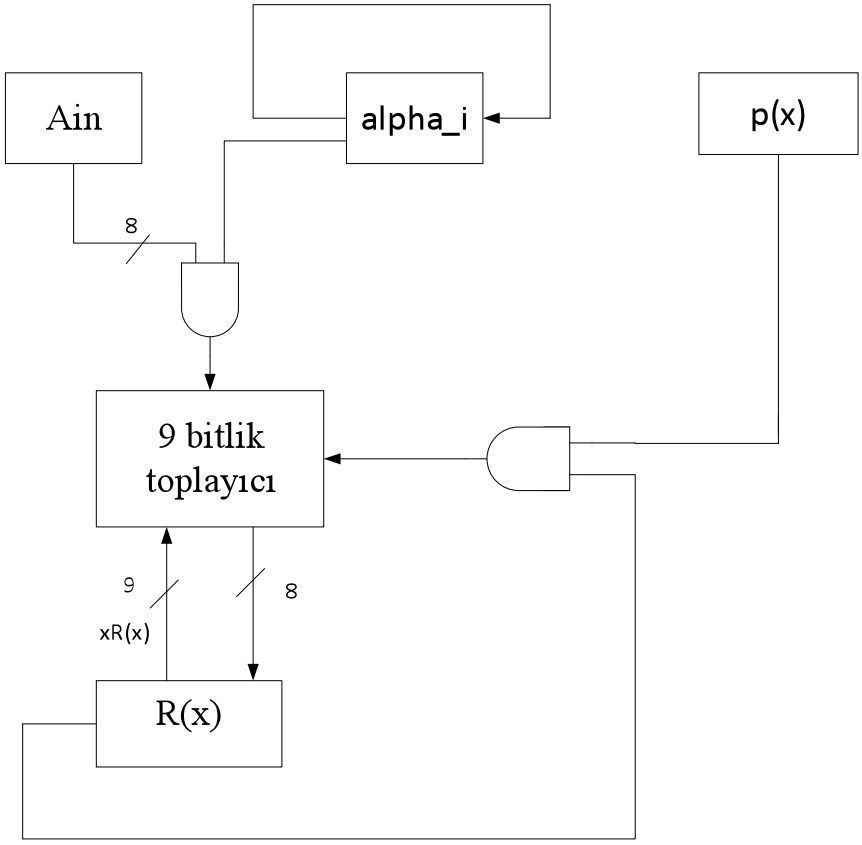
Şekil 5.4.a: Çarpma Bloğu Birim Hücresi



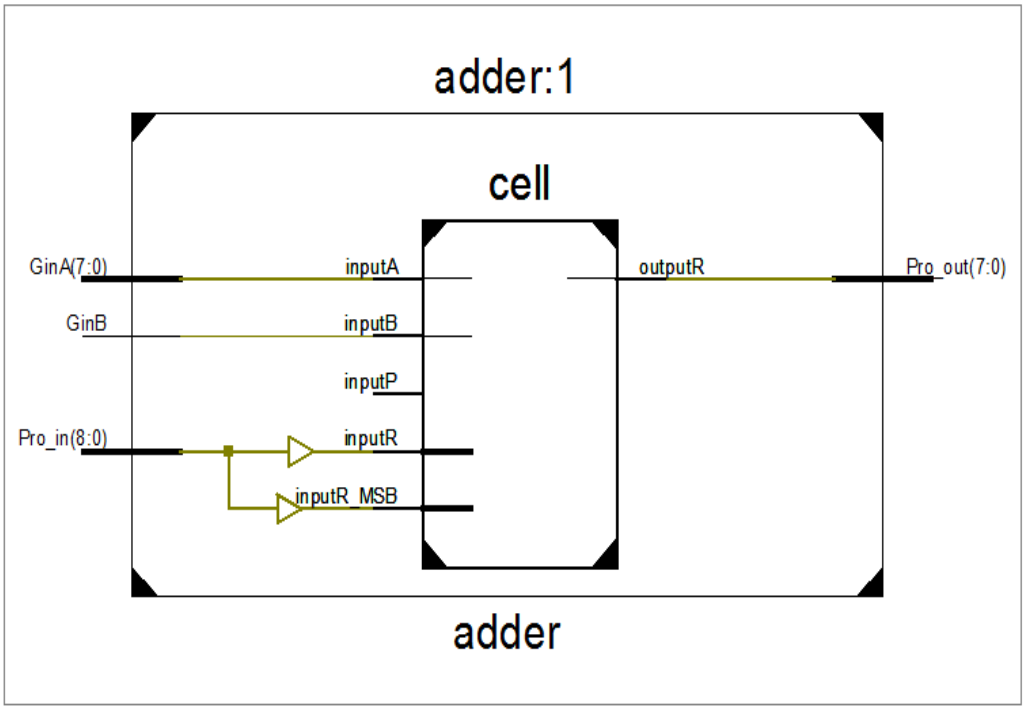
Şekil 5.4.b: Çarpma Bloğu Birim Hücresinin RTL Şeması

Şekil 5.4.b de görüldüğü üzere, çarpma bloğu birim hücresinin RTL (Register Transfer Level) şemasında görüldüğü üzere A sayısının bitleri inputA ile alpha_i

sayısının bitleri inputB ile gösterilmiştir. inputR ve inputP de srası ile R sayısının bitleri ve p sabitinin bitleri için kullanılmıştır.



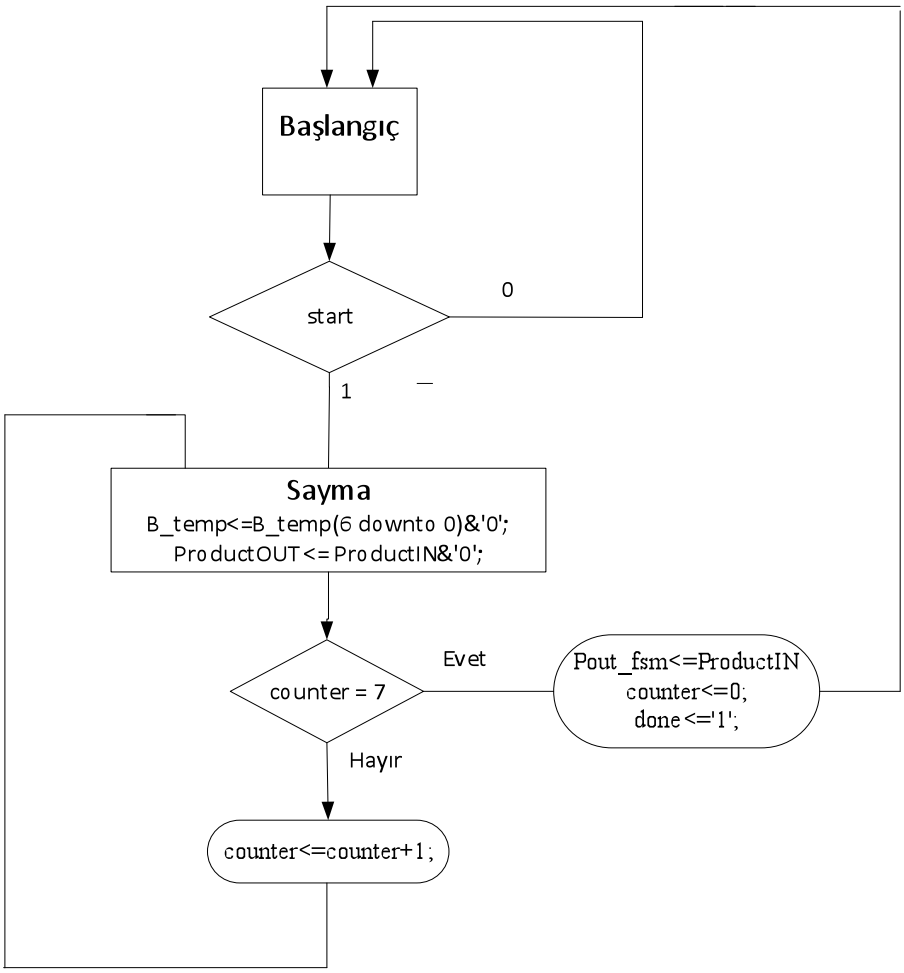
Şekil 5.5.a: Çarpma Bloğu Blok Diyagramı



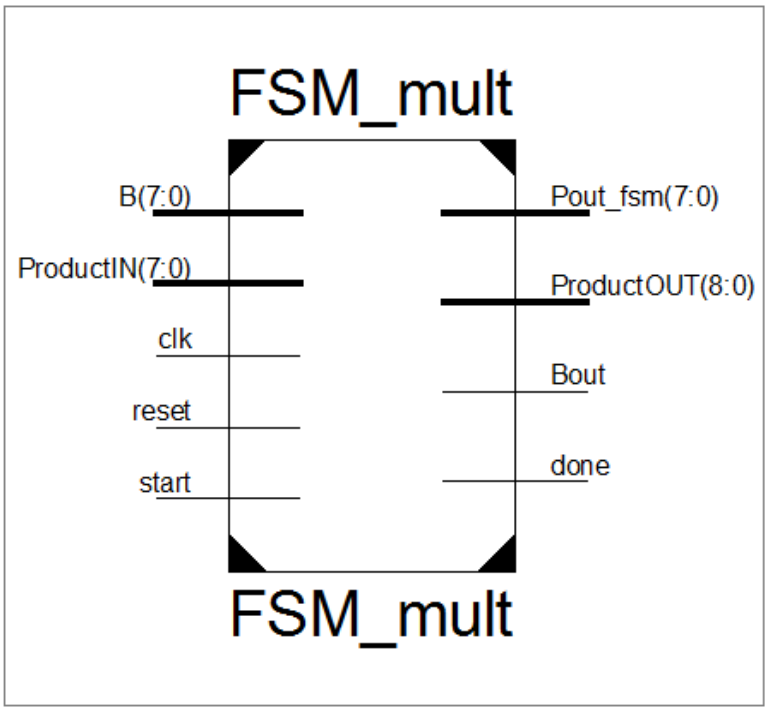
Şekil 5.5.b: Çarpma Bloğu Blok Diyagramının RTL Şeması

Şekil 5.5.a da R girişinin 1 basamak sola kaydırılmasıyla elde edilen 9 bitlik sayı Pro_in girişiyle gösterilmiştir. A sayısı B sayısının 1 biti ile çarpılmış ve GF(2^8) alanının indirgenemez polinomu $x^8+x^4+x^3+x^2+1$ ile belirtilen "100011101" sayısı adder bloğu içinde sabit (constant) olarak tanımlanmıştır. Şekil 5.5.b de ise birim hücrelerin bir araya getirilmesiyle elde edilen VHDL de gerçeklenen çarpma bloğunun RTL şeması verilmiştir.

Tasarlanan çarpma devresindeki amaç 8 bitlik 2 sayının (A ve alpha_i) GF(2^8) alanında çarpılmasıdır. Şekil 5.5.b de çarpma bloğu devresi A sayısını (GinA) giriş olarak alır ve alpha_i nin bir biti için (GinB) çarpma işlemini yapar. Şekil 5.6.a da ise toplayıcı devresine alpha_i nin bitlerini teker teker göndermek için bir sonlu durum makinası tasarlanmıştır. FSM de çarpma devresinin alpha_i girişi B ile gösterilmiş ve B nin bitlerini toplayıcıya sırayla gönderen bir sayıcı (counter) tanımlanmıştır. B 8-bitlik bir sayı olduğu için sayıcı 0 dan başlayıp birer artırılarak 7 ye ulaştığında çarpma işlemi biter ve başlangıç durumuna dönülür. Çarpıcı bloğundaki FSM başlangıç ve sayma durumu olmak üzere 2 duruma dayalıdır.

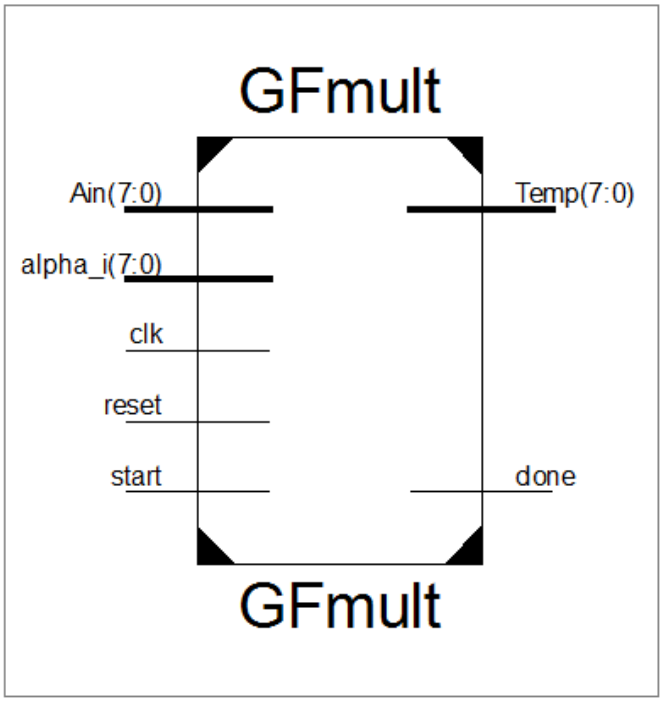


Şekil 5.6.a: Çarpma İşlemi İçin Oluşturulan FSM

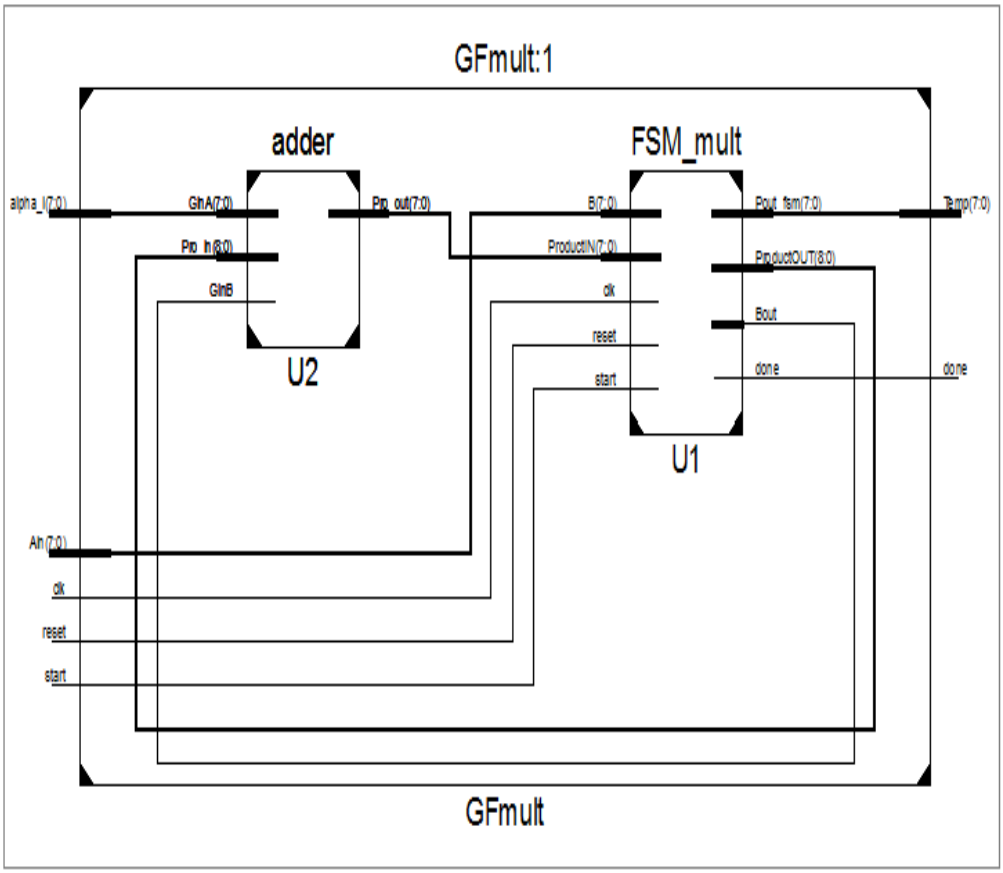


Şekil 5.6.a: Çarpma İşlemi İçin Oluşturulan FSM nin RTL Şeması

Şekil 5.6.b de ise çarpıcı için tasarlanan FSM nin RTL si verilmiştir. Şekil 5.6.a da görülen ve FSM bloğu içinde sinyal olarak tanımlanan B_temp, B girişinin bitlerini teker teker çıkıştaki Bout bitine atama işlemini yapar.



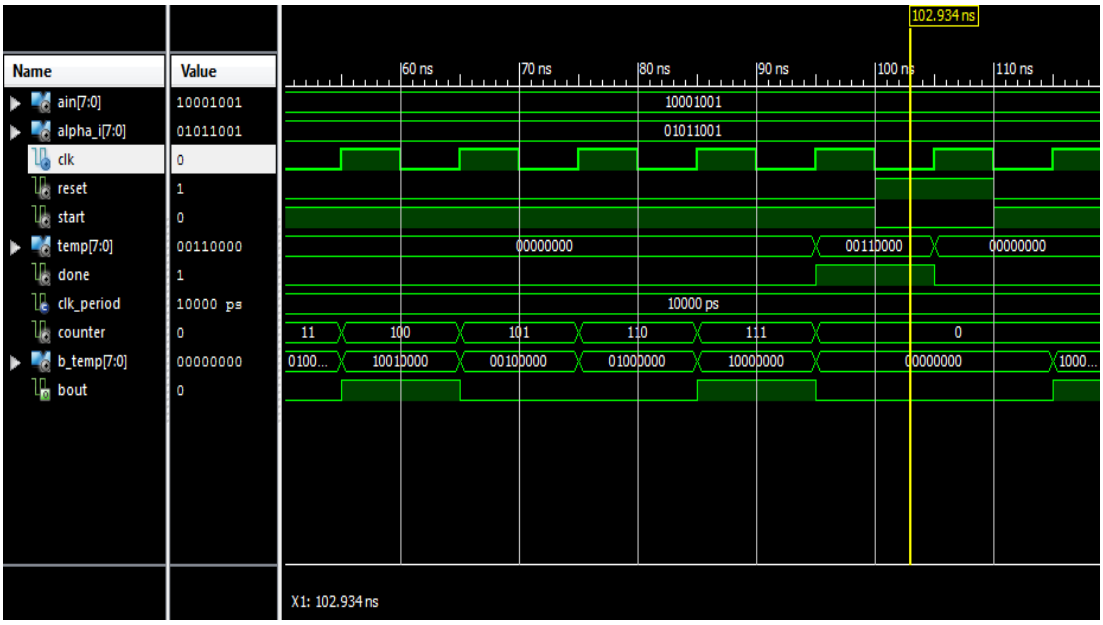
Şekil 5.7.a: Sonlu Alanda 8-bitlik Çarpıcının RTL Şeması



Şekil 5.7.b: Çarpıcı Bloğunun İç Yapısının RTL Şeması

Şekil 5.7.b den görüldüğü üzere, toplayıcı devresinin Pro-out çıkışı ile FSM devresinin ProductIN girişi ve FSM devrensin ProductOUT çıkışı ile toplayıcı devresinin Pro_in girişi birbirine bağlıdır. Bir anlamda, FSM toplayıcı devresindeki işlemleri kontrol ederek işlem bittiğinde sonucun çarpma devresinin çıkışından alınmasını sağlar.

Şekil 5.8 de GF çarpıcının simülasyon sonucu Bölüm 3.2.2 deki örnek için verilmiştir. Ain ve alpha_i girişleri sırasıyla 10001001 ve 01011001 alınarak Bölüm 3.2.2 de gösterildiği gibi Temp çıkışı yani çarpma işleminin sonucu 00110000 olarak görülmektedir.

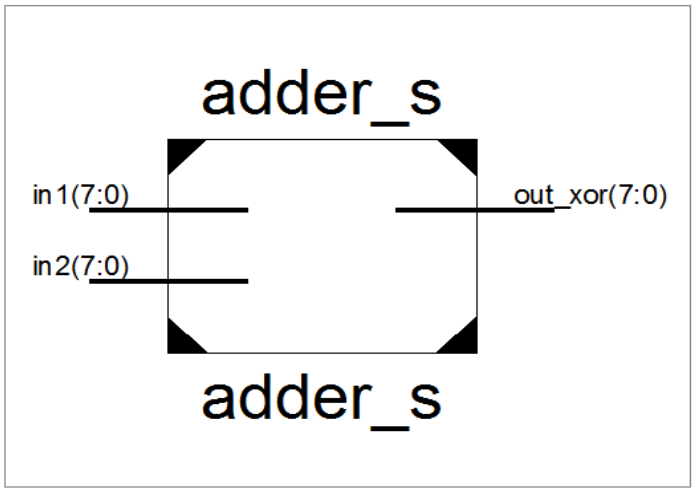


Şekil 5.8: Tasarlanan Çarpıcının Simülasyon Sonucu

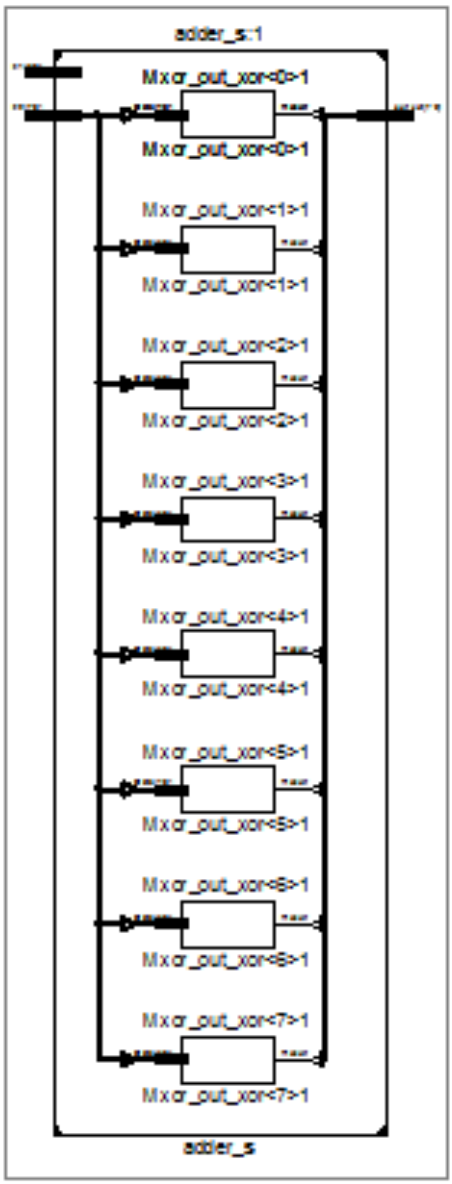
5.1.2. Toplama Bloğu

Sonlu alanda toplama işlemi, devrenin girişine gelen sayıları gösteren polinomların aynı dereceli terimlerinin XOR işlemine alınması anlamına gelmektedir.

A(x)=$\sum_{i=0}^{7} a_i x^i$ ve B(x)=$\sum_{i=0}^{7} b_i x^i$, GF(2^8) de sayıları temsil eden polinomlar ise; toplamları C(x) = $\sum_{i=0}^{7} (a \oplus b_i)\, x^i$ olur.



Şekil 5.9.a: Sendrom Hesaplamada Kullanılan Toplama Bloğunun RTL Şeması



Şekil 5.9.b: Toplama bloğundaki XOR işlemlerini gösteren RTL şeması

Şekil 5.9.a ve 5.9.b de sendrom hesaplamada kullanılan toplama bloğunun RTL şemaları verilmiştir. Bu blok Şekil 5.2 deki sendrom hesaplama devresinden görüldüğü üzere toplama devresinde çarpma devresinin çıkışı ve R_i nin XOR lanması işleminde kullanılmaktadır.

5.2. Anahtar Eşitlik Çözme Bloğu

İlk blokta hesaplanan sendromlar bir sonraki Anahtar Eşitlik Çözücü (Key Equation Solver-KES) bloğunda;

$$S(x)\sigma(x) = \omega(x) \bmod x^{2t} \qquad (5.4)$$

anahtar eşitliğinin çözümünde kullanılacaktır [15].

S(x) polinomundan hareketle σ(x) ve w (x) polinomlarının belirlenmesi kod çözme sürecinin en zor aşamasıdır[17]. Aynı zamanda Reed–Solomon kod çözücünün blokları arasında anahtar eşitlik çözücü bloğu donanım karmaşıklığının en yüksek, gecikme miktarının en fazla olduğu bloktur. Bu yüzden 5.4 eşitliğinin çözümünde ihtiyaçların gerçek zamanlı olarak karşılanması ve donanım karmaşıklığının azaltılması için birçok yöntem geliştirilmiştir [18].

Anahtar eşitliğinin çözümünde genellikle Berlekamp–Massey (BM) [19] ve modifiye edilmiş Euclid (modified Euclid-ME)[16] algoritmaları kullanılmaktadır. Euclid ve modifiye edilmiş Euclid algoritmaları karmaşık bir veri yolu gerektirdiğinden ve daha fazla donanım gerektirdiğinden Değişmeden Yeniden Formüle Edilmiş Berlekamp Massey (Reformulated inversionless Berlekamp Massey-RiBM) algoritması geliştirilmiştir. RiBM algoritması anahtar eşitlik çözücü bloğundaki gereksiz bileşenleri kaldırır, böylelikle daha az donanım karmaşıklığı ve daha az gecikme süresi sağlanmış olur [17]. Tablo 5.2 de ME ve RiBM algoritmalarının performans karşılaştırması verilmektedir.

Tablo 5.2: ME ve RiBM Algoritmalarının Performans Karşılaştırması [18].

	ME	RiBM
Hücre sayısı	$2t$	$3t+1$
Kaydedici	$50t$	$6t+2$
Multiplexer	$38t$	$3t+1$
Çarpıcı	$8t$	$6t+2$
Toplayıcı	$4t$	$3t+1$
Gecikme	$2t$	$2t$

Bu tezde, 5.4 ile gösterilen anahtar eşitliğinin çözülmesi için RiBM algoritması kullanılmıştır.

Gerçeklemede Kullanılan RiBM algoritması [18]:

Başlangıç:

$\delta_{3t}(0) = 1$; $\delta_i(0) = 0$; for $i = 2t, 2t+1, \ldots, 3t-1$.

$k(0) = 0$; $y(0) = 1$.

Giriş: S_i, $i = 0,1, \ldots, 2t-1$.

$\delta_i(0) = \theta_i(0) = S_i$, $(i = 0,1, \ldots, 2t-1)$

for $r = 0$ step 1 until $2t-1$ do

begin

Adım RiBM.1 $\delta_i(r+1) = \gamma(r) \cdot \delta_{i+1}(r) - \delta_0(r) \cdot \theta_i(r)$,

$(i = 0,1, \ldots, 3t)$

Adım RiBM.2

if $\delta_0(r) \neq 0$ and $k(r) \geq 0$ then

begin

$\theta_i(r+1) = \delta_{i+1}(r)$, $(i = 0,1, \ldots, 3t)$

$\gamma(r+1) = \delta_0(r)$

$k(r+1) = -k(r) - 1$;

end

else

begin

$\theta_i(r+1) = \delta_i(r)$, $(i = 0,1, \ldots, 3t)$

$\gamma(r+1) = \gamma(r)$

$k(r+1) = -k(r) - 1$;

end

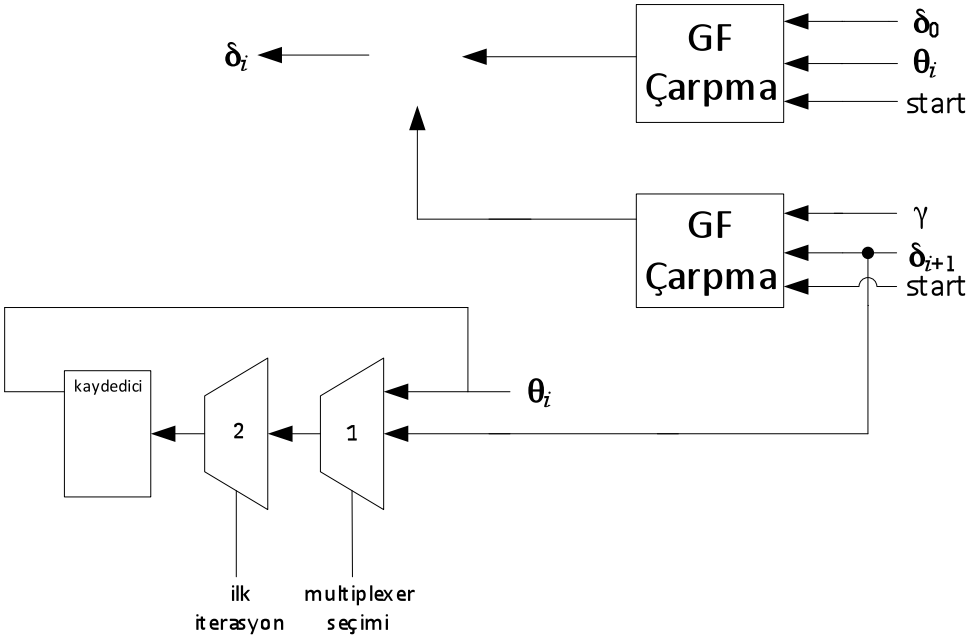
end

Çıkış:

$\lambda_i(2t) = \delta_{t+i}(2t)$, $(i = 0,1, \ldots, t)$

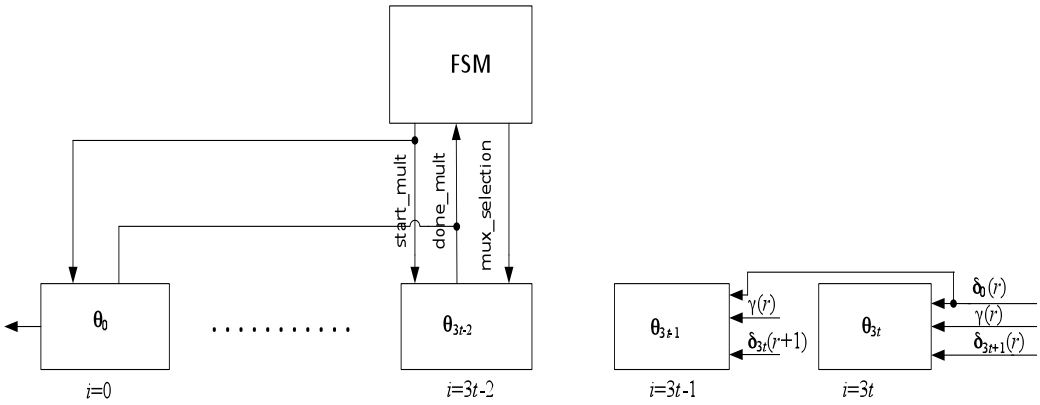
$R_i(2t) = \delta_i(2t)$, $(i = 0,1, \ldots, t\text{-}1)$

Yukarıdaki algoritmadaki 1. adımdan görüldüğü üzere RiBM bloğu $3t$+1 (t=8 olduğundan $3t$+1=25) tane temel RiBM hücresi gerçeklenmiştir. 1 tane RiBM hücresi için blok şema Şekil 5.10 da verilmiştir.

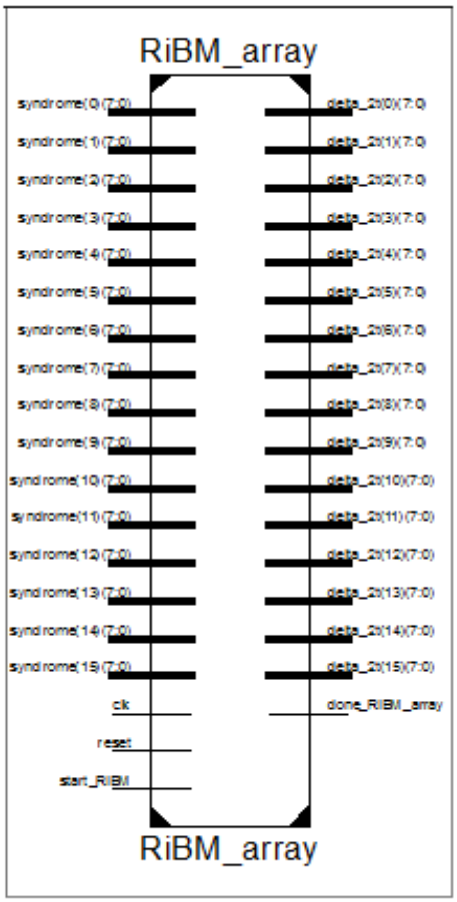


Şekil 5.10: RiBM algoritması birim hücresi

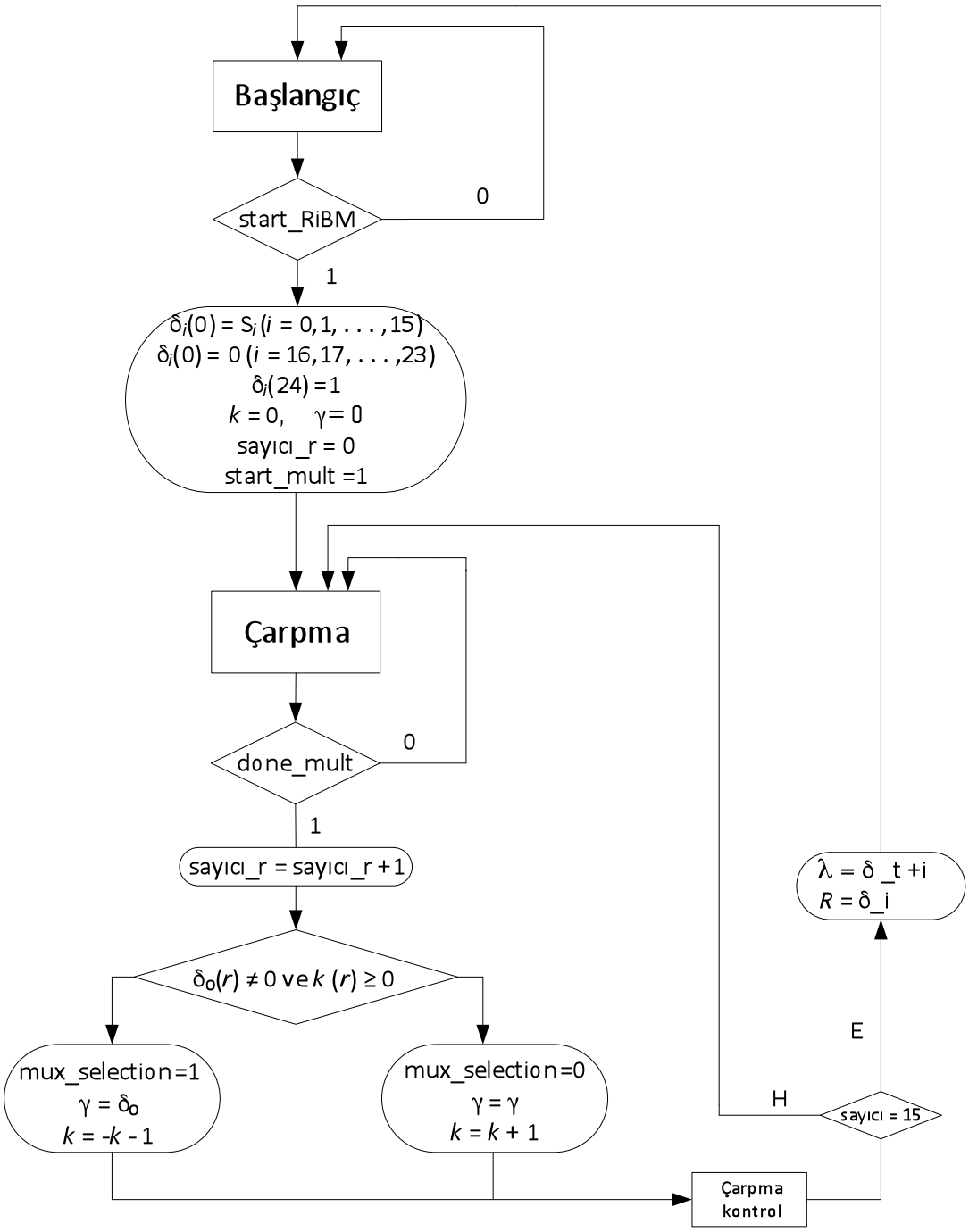
RiBM algoritmasının 1. adımında görüldüğü üzere $\delta_i(r+1) = \gamma(r) \cdot \delta_{i+1}(r) - \delta_0(r) \cdot \theta_i(r)$ işleminin yapılabilmesi için 2 tane GF çarpıcı bloğuna ve 1 adet XOR bloğuna ihtiyaç vardır. Şekil 5.11.a da 25 adet RiBM hücresi tasarlanan bir FSM yardımıyla bir araya getirilmiştir. Şekil 11.b de ise RiBM algoritmasının gerçeklemesinin RTL şeması verilmiştir. Şekil 5.12 de bu hücrelerin çalışmasının bilgisini taşıyan FSM nin blok diyagramı verilmiştir.



Şekil 5.11.a: RiBM Algoritması Blok Şeması



Şekil 11.b: RiBM algoritması için RTL şeması



Şekil 5.12: RiBM algoritmasının gerçeklenmesi için oluşturulan FSM

6. SONUÇLAR VE TARTIŞMA

Bu bitirme çalışmasında gürültülü bir iletişim kanalında güvenli iletişimin sağlanabilmesi için kullanılan hata belirleme ve düzeltme teknikleri ile ilgili bilgi verilmiştir. İletişim kanalından geçen bilgiye eklenen fazladan bilgi sembolleri hata olarak adlandırılır ve bir kod çözücünün amacı bu fazladan eklenen sembolleri kullanarak verinin ilk halini elde etmektir. Bu çalışmada hata düzeltme tekniklerinin önemli bir sınıfı olan Reed-Solomon kodlar detaylı olarak anlatılmış ve Reed-Solomon kod çözücü VHDL yardımıyla gerçeklenmeye çalışılmıştır.

Reed-Solomon kodlar hata düzeltme kodlaması tekniklerinden ikili olmayan BCH kodları sınıfına dahildir ve bu kodların matematiksel temelleri sonlu alana dayanmaktadır. Dolayısıyla bu kodlama bloklarındaki çarpma ve toplama işlem blokları sonlu alandaki çarpma ve toplamadır. Sonlu alanda çarpma ve toplama işlemleri VHDL de gerçeklenmiş ve bu işlemlerin gerektiği tüm bloklarda bunlar kullanılmıştır.

Gerçeklemeler RS(255,239) koduna göre yapılmıştır ve gerçeklenen blokların olabildiğince küçük alan kaplaması ilk hedef olmuştur. Kod çözücünün blok şeması hakkında bilgi verilmiş ve sendrom hesaplama ve anahtar eşitlik çözücü bloğu bu çalışmada gerçeklenmiştir. Forney ve Chien araştırmaları ve hata düzeltme blokları, Seyyid M Dilek'in yüksek lisans tezinden alınmıştır [3].

KAYNAKLAR

[1] **Reed, I. S. and Solomon, G.,** Polynomial codes over certain finite fields," *SIAM Journal of Applied Math.*, vol. 8, 1960, pp. 300-304.

[2] **Robert, H.M. and Zaragoza,** 2002. The Art of Error Correcting Coding, John Wiley & Sons, Chichester.

[3] **Dilek, S.M.,** 2013. DVB-S alıcısı için ileri hata düzeltme birimi gerçeklemesi, *Yüksek Lisans Tezi*, İ.T.Ü. Fen Bilimleri Enstitüsü, İstanbul.

[4] **Glover, I.A. and Grant, P.M.,** 2010. Digital Communications, Prentice Hall, Harlow.

[5] **Sweeney, P.,** 1991. Error Control Coding: an introduction, Prentice Hall International, N.J.

[6] **Lin, S. and Costello, D.J.,** 2004. Error Control Coding, Pearson Prentice Hall, N.J.

[7] **Sklar, B. and Harris J.H.,** 2004. The ABCs of linear block codes, *IEEE Signal Processing Magazine*, **04**, 15.

[8] **Sklar, B.**, 2001. Digital Communications: Fundamentals and Applications, Prentice Hall, N.J.

[9] **Bartee, T.C. and David, I.S.**, 1963. Computation with finite fields, *Information and Control*, **6**, 79-98.

[10] **Moreira, J.C. and Farrell P.G.**, 2006., Essentials of Error Control Coding, John Wiley & Sons, Chichester.

[11] **Forney, G.D.,** On decoding BCH codes, *IEEE Transactions on Information*

Theory, vol.IT-11, No. 4, pp.549-557, October 1965.

[12] **H.O. Burton and D.D. Sullivan,** Errors and error control, Proc. IEEE, 60(11): 1293-1310, November 1972.

[13] **Singleton, R.C.,** Maximum distance q-nary codes, *IEEE Transactions on Information Theory*, vol. IT-10, no. 2, pp. 116–118, Apr. 1964.

[14] **Fossorier, M.P.C., Mihaljevic, M. and Imai, H.,** Reduced complexity iterative decoding of low-density parity-check codes based on belief propagation, *IEEE Trans. Commun.*, vol. 47, pp. 673–680, May 1999.

[15] **Park, J., Lee, K., Choi, C. and Lee., H.,** 2009. High-speed low-complexity Reed-Solomon decoder using pipelined Berlekamp-Massey algorithm, *IEEE ISOCC*, pp.452-455.

[16] **Lee, H., Yu, M. And Song, L.,** 2000. VLSI design of Reed–Solomon decoder architectures, *IEEE International Symposium on Circuits and Systems*, Geneva, Switzerland, May 28-31.

[17] **Sarwate, D.V. and Shanbhag, N.R.,** High-speed architectures for Reed-Solomon decoders. *IEEE Transactions on VLSI Systems*, vol.9, pp. 641-655, October 2001.

[18] **Baek, J. and Sunwoo, M.H.,** Low hardware complexity key equation solver chip for Reed-Solomon decoders, *IEEE Asian Solid-State Circuits Conference*, Jeju, Korea, November 12-14.

[19] **A. Raghupathy and K. J. R. Liu,** Algorithm-based low-power / high-speed Reed-Solomon decoder design, *IEEE Trans. Circuit Syst. II*, vol. 47, pp. 1254-1270, Nov. 2000.

ÖZGEÇMİŞ

Gizem Kaçmaz, 1991 yılında Bursa'da doğdu. Lise eğitimini Bursa Ulubatlı Hasan Anadolu Lisesi'nde aldı. Ardından 2009 yılında İstanbul Teknik Üniversitesi Elektronik ve Haberleşme Mühendisliği Bölümü Elektronik Mühendisliği Programı'nda lisans eğitimine devam etti. Temel ilgi alanı sayısal doananım tasarımıdır.