Very Compact FPGA Implementation of the AES Algorithm

Term Project

Zeynep Delal Mutlu[1], Dr. Tamer Güdü[2]

[1]Computer Engineering/Cyber Security,Tobb University,
Söğütözü, Söğütözü Cd. No:43, 06510 Çankaya, Ankara, Turkey

[2]Electrical and Electronics Engineering, Tobb University,
Söğütözü, Söğütözü Cd. No:43, 06510 Çankaya, Ankara, Turkey

[2]Lead FPGA Engineer, Aselsan,
Yenimahalle, Ankara, Turkey

[1]*zmutlu@etu.edu.tr,* [1]*zeynepdelalmutlu@alumni.bilkent.com.tr,* [1]*zeynepdelalmutlu@gmail.com*

[2]*gudu.tamer@ttmail.com,* [2]*gudu.tamer@gmail.com*

[1,2]*www.etu.edu.tr,* [2]*www.aselsan.com.tr*

ABSTRACT

In this page, very compact FPGA implementation of AES encryption and decryption algorithms with 128-bit key study is targeted. The ultimate aim is practising low-cost embedded applications for both encryption and decryption of AES 128-key algorithms within the same hardware design. Hardware design of the compact encryption and decryption algorithm, key schedule design and their implementations are discussed in detail. Regarding the compact design, the conversions between the encryption algorithm steps and their reverse steps for the decryption algorithm are investigated and shared resources of these steps are presented. At the end of the study, there are reviews of why this design was successful.

Tobb Universitesi, Fen Bilimleri Fakültesi
Türkiye, Ağustos 2020

1. GİRİŞ

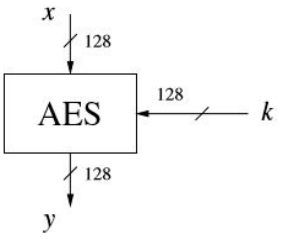
Amerika Birleşik Devletleri federal devletinin standart enstitüsü olan Ulusal Standartlar ve Teknoloji Enstitüsü(NIST), 2001 yılında AES(Advanced Encryption Standart) algoritmasını ulusal şifreleme standardı olarak seçmiştir. Bu algoritmanın 128-bit, 196-bit ve 256-bit olmak üzere 3 farklı versiyonu mevcuttur. Ulusal standart seçildiği günden itibaren tüm dünyada birçok devlet tarafından benimsenmiş ve sahada uygulanmaya başlanmıştır.

AES algoritması günümüze kadar hem yazılım hem de donanım bazlı platformlarda, birçok farklı yazılım dilinde yazılmış ve farklı cihazlarda gerçeklenmiştir. Her yazılım dilinin ve donanımın farklı gereksinimleri mevcuttur. Bu durum, AES algoritmasının pratiğe dökülmesi esnasında farklı teknik adımların uygulanmasına imkan vermektedir. Şimdiye kadar, dünya genelinde birçok akademisyen ve çalışan grubu farklı yöntemlerle AES algoritmasının implementasyonunu gerçekleştirmeyi başarmıştır.

Bu çalışmaların birbirlerine benzeyen ve birbirlerinden ayrılan noktaları vardır. Bu makalenin konusu olan akademik çalışmada, AES 128-bit algoritmasının hem şifreleme hem de-şifreleme yöntemlerinin aynı donanım dizaynı üzerinde olabildiğince kompakt bir şekilde gerçeklenmesi amaçlanmıştır. Bu çalışmada esas alınan donanım Xilinx Spartan II XC2S30 FPGA cihazıdır. Algoritmaların yazıldığı dil ise VHDL yazılım dilidir.

2. AES ALGORİTMASI

AES algoritması; şifreleme, deşifreleme ve anahtar üretimi olmak üzere üç ana işlem çerçevesinde değerlendirilmektedir. Bu makalenin konusu olan 128-bit AES şifreleme algoritması, 128-bit düz metin ve 128-bit anahtar girdilerini alır ve 128-bit şifreli metin çıktısı verir. Aynı şekilde düşünecek olursak; 128-bit AES deşifreleme algoritması da, 128-bit şifreli metin ve 128-bit anahtar girdilerini alır ve 128-bit şifreli düz çıktısı verir(Şekil-1).

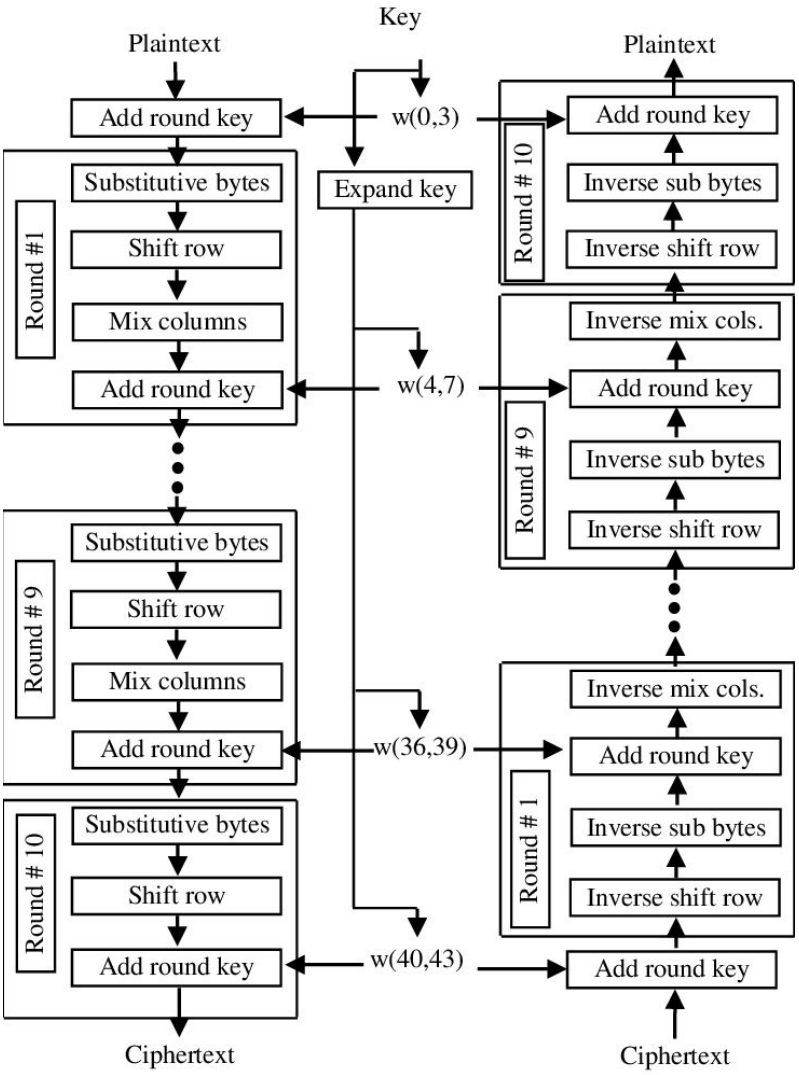


Şekil 1. *AES 128-bit şifreleme parametreleri(girdi, çıktı ve anahtar)*

2.1 NIST Standardı AES 128-bit Şifreleme ve Deşifreleme Algoritması

AES 128-bit şifreleme algoritması 10 round'dan oluşmaktadır. Her roundda işleme giren ve 128-bit anahtardan türetilmiş olan alt anahtarlar vardır. Aynı alt anahtarlar, deşifreleme algoritmasında sondan başlanarak kullanılırlar. Bunun sebebi ise; deşifreleme algoritmasının şifreleme algoritmasına göre sondan başa doğru gerçekleştirilmesidir. Örneğin; şifreleme algoritmasının ilk roundu, deşifreleme algoritmasında son rounda denk gelmektedir(Şekil-2).

Ayrıca hem şifreleme hem de deşifreleme algoritmaları Addroundkey işlemi ile başlamaktadır. Daha sonra 10 round boyunca Şekil-2'de belirtilen işlemler yapılır. Her iki algoritmada da son roundda Mixcloumn işlemi yapılmaz.



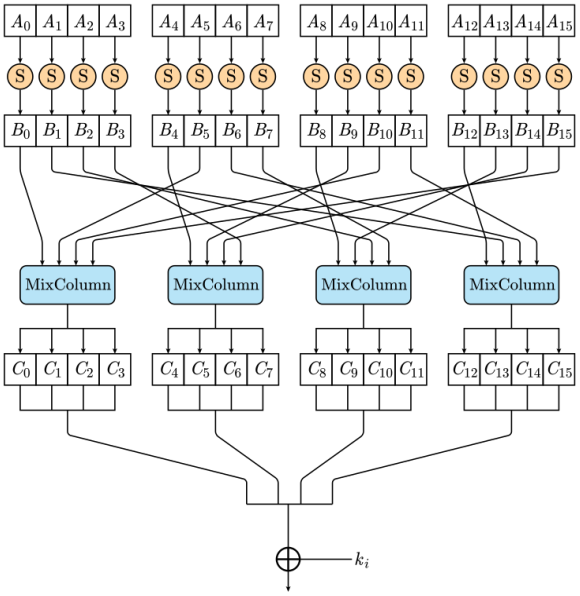
Şekil 2. *AES 128-bit şifreleme ve deşifreleme algoritmaları*

Şifreleme algoritmasının roundlarında sırasıyla Subbyte, Shiftrow, Mixcolumn ve Addroundkey işlemleri yapılır. Deşifreleme algoritmasında ise bu işlemlerin tersi yapılır; InvSubbyte, InvShiftrow, InvMixcolumn ve Addroundkey. Addroundkey işlemi sadece "xor" işlemi içerdiği için, bu işlemin tersi(inverse of Addroundkey) yine kendisidir.

AES 128-bit şifreleme ve deşifreleme algoritmalarının çıktılarının her yazılım dili ve her donanımda aynı şekilde elde edilmesi gerekir. Bunun için hiçbir işlem basamağı atlanmamalı ve farklılaştırılmamalıdır.

Şekil-2'deki AES 128-bit şifreleme algoritmasının bir roundunun detayları Şekil-3'te mevcuttur.

Şekil-3'te sırasıyla Subbyte, Shiftrow, Mixcolumn ve Addroundkey işlemleri yapılmaktadır. Tüm bu işlemleri gerçekleştirebilmek için 128-bit düz metin 8-bit(byte)'lik parçalara bölünmektedir. İşlemler ve yer değiştirmeler 8-bitlik parçalar baz alınarak yapılmaktadır. En sonunda ise Addroundkey işleminde 32-bit'lik parçalar halinde işlem yapılması için 4 adet 8-bit bir araya getirilmekte ve 32-bit'lik parçalar elde edilmektedir. Bu işlemi tüm Mixcolumn çıktılarını birleştirip tek bir parça 128-bit çıktı ile 128-bit alt anahtarı işleme sokuyormuş gibi düşünmek de mümkündür.



Şekil 3. *AES 128-bit şifreleme algoritmasında bir round*

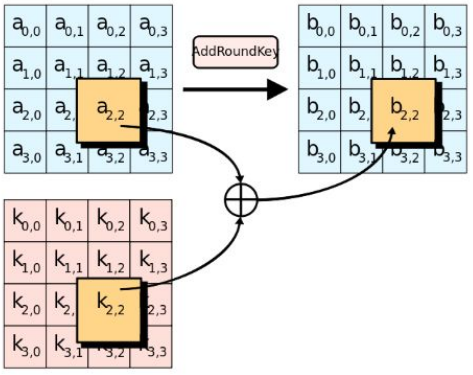
2.2 NIST Standardı AES 128-bit İşlem Basamakları

AES şifreleme algoritmasında Addroundkey, Subbyte, Shiftrow ve Mixcolumn işlemleri kullanılırken; AES-128 bit deşifreleme algoritmasında bu işlemlerin ters fonksiyonları yani Addroundkey, InvSubbyte, InvShiftrow ve InvMixcolumn işlemleri kullanılır.

AES 128-bit şifreleme ve deşifreleme algoritması, 128 bitin tamamının, byte'lar(8-bit) halinde bir matris üzerine yerleştirilmesi ile işleme alınır. Bu matris, dört satır ve dört sütundan oluşur. Byte'ların matrise dizilişi, sütunlar halinde, yukarıdan aşağı doğru olur. İlk dört byte ilk sütuna ve ikinci dört byte ikinci sütuna yazılarak devam eder.

2.2.1 Addroundkey

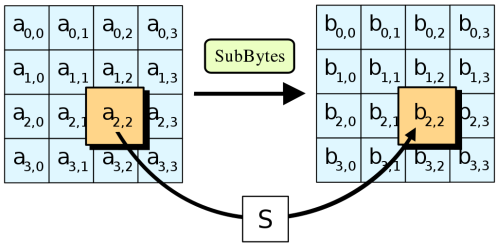
Addroundkey işlemi, Şekil 4'te de görüldüğü üzere 128 bitlik input sonucu ile 128 bitlik alt anahtarını "xor" işlemine sokar. "Xor" işleminin ters fonksiyonu yine kendisi olduğu için hem şifreleme de hem de deşifrelemede Addroundkey işlemi aynı şekilde kullanılır. Dolayısıyla bu işlem adımının tersine ihtiyaç duyulmamıştır. AES algoritmasında işlemler byte bazında işleme alındığı için, Şekil 4'teki şema Addroundkey işlemini temsil eder.



Şekil 4. *AES 128-bit şifreleme ve deşifreleme Addroundkey işlemi*

2.2.2 *Subbyte ve InvSubbyte*

Subbyte işleminde, girdinin bir byte'ı S-box'ta işleme girer. Sonuç olarak S-box'ın o byte için belirlediği sabit değer atanır(Şekil 5). S-box'a giren her byte, sabit bir byte'a karşılık geldiği için tüm bu byte karşılıklarını bir matris olarak düşünebiliriz(Şekil 6).



Şekil 5. *AES 128-bit Subbyte işlemi*

Bir byte 8 bitten oluşur ve iki adet hexadecimal değer ile ifade edilir. İlk hex değeri Tablo 1'de 'x' değerini, ikinci hex değeri ise tablodaki 'y' değerini ifade eder. Kısaca Subbyte işlemi, byte'ın tablodaki karşılığıdır.

x \ y	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Tablo 1. *AES 128-bit Subbyte Tablosu*

Deşifreleme işleminde kullanılan InvSubbyte tablosu ise tıpkı şifreleme işlemindeki Subbyte tablosu gibi oluşturulur ve Subbyte tablosunun tersidir(Tablo 2).

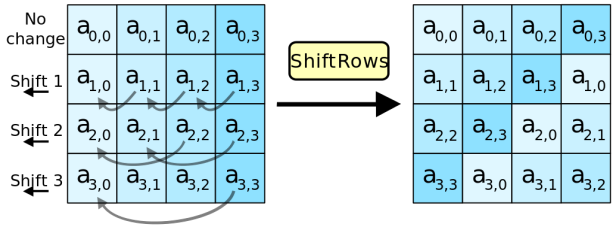
x \ y	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
5	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
6	90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
A	47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
B	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
C	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
D	60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
E	A0	E0	3B	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
F	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

Tablo 2. *AES 128-bit InvSubbyte Tablosu*

Tobb Universitesi, Fen Bilimleri Fakültesi
Türkiye, Ağustos 2020

2.2.3 *Shiftrow ve InvShiftrow*

Shiftrow işlemi, AES-128 şifreleme algoritmasında kullanılan bir işlemdir. Girdi olarak oluşturulan matriste ilk satır aynen kalır. İkinci satır bir birim, üçüncü satır iki birim ve son satır ise 3 birim sola kaydırılarak çıktı matrisi elde edilir(Şekil 6).

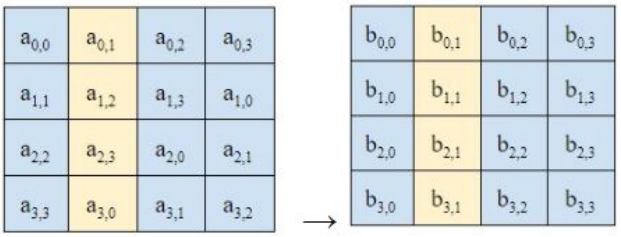


Şekil 6. *AES 128-bit Shiftrow İşlemi*

Deşifreleme algoritmasında Shiftrow işleminin tersi olan InvShiftrow işlemi kullanılır. Girdi olarak şifreli metin marisi kullanılır. İlk satır aynen kalır. İkinci satır bir birim, üçüncü satır iki birim ve son satır ise 3 birim sağa kaydırılarak çıktı matrisi elde edilir.

2.2.4 *Mixcolumn ve InvMixcolumn*

AES şifreleme algoritmasında kullanılan Mixcolumn işlemi dört byte yani 32 bit ile işleme girer. Sonuç olarak yine 32 bitlik çıktı verir(Şekil 3). Mixcolumn, Shiftrow işleminden sonra uygulandığı için kaydırma işlemi gerçekleştirilmiş sütun değerleri geçerlidir. O yüzden



Şekil 7. *AES 128-bit Mixcolumn İşlemi*

Şekil 7'de girdi sütunu Şekil 8'deki matris ile çarpma işlemine girmiş ve sonuç olarak Şekil 7'deki sonuç sütunu elde edilmiştir.

$$\begin{pmatrix} b \end{pmatrix} = \begin{pmatrix} 02\ 03\ 01\ 01 \\ 01\ 02\ 03\ 01 \\ 01\ 01\ 02\ 03 \\ 03\ 01\ 01\ 02 \end{pmatrix} \begin{pmatrix} a \end{pmatrix}$$

Şekil 8. *AES 128-bit Mixcolumn Matris Çarpma İşlemi*

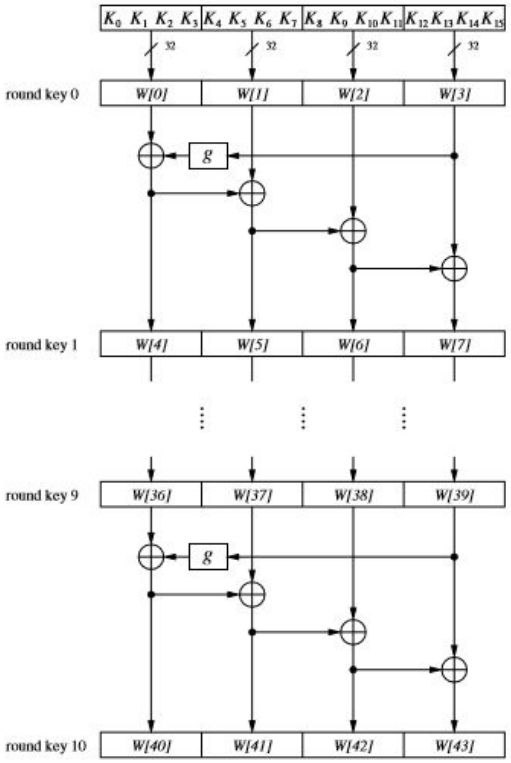
Şekil 9'da gerçekleştirilen InvMixcolumn işlemi için, Şekil 8'deki çarpım matrisinin tersi kullanılmaktadır.

$$\begin{pmatrix} a \end{pmatrix} = \begin{pmatrix} 0E\ 0B\ 0D\ 09 \\ 09\ 0E\ 0B\ 0D \\ 0D\ 09\ 0E\ 0B \\ 0B\ 0D\ 09\ 0E \end{pmatrix} \begin{pmatrix} b \end{pmatrix}$$

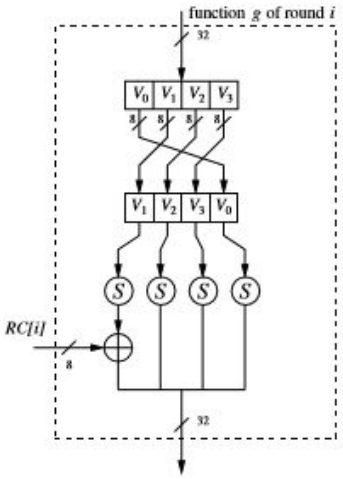
Şekil 9. *AES 128-bit InvMixcolumn Matris Çarpma İşlemi*

2.3 NIST Standardı AES 128-bit Alt Anahtar Üretimi

Alt anahtarlar üretmek için Şekil-10'te yer alan algoritma uygulanmaktadır. Bu algoritmada görülen g() fonksiyonunun temel mantığı Şekil-11'teki gibidir.



Şekil 10. *AES 128-bit alt anahtar üretimi*



Şekil 11. *AES 128-bit alt anahtar üretiminde yer alan g() fonksiyonu*

Alt anahtar üretimi için 128-bit anahtar 32-bit dört parçaya ayrılır. Hem encryption hem de decryption algoritmasının en başında uygulanan Addroundkey işlemi için bu anahtarlar hiçbir değişikliğe uğramaz. Sonraki 10 round için işlem gören 128-bit altanahtarlar uygulanır. Toplamda 11 adet 128-bitlik alt anahtar elde edilir.

Şekil 11'te yer alan ve yalnızca bir adet 8-bit'lik parçaya uygulanan "xor" işlemi için RC[i] ile gösterilen değerlere sahip olunmalıdır. Bu değerler, AES standardında belirtildiği üzere, her round için sabit değerleri ifade etmektedir ve round sayıları şekildeki 'i' değeriyle ifade edilmektedir(Tablo 3).

i	1	2	3	4	5	6	7	8	9	10
Hex değeri	01	02	04	08	10	20	40	80	1B	36

Tablo 3. *AES alt anahtar üretiminde yer alan g() fonksiyonundaki RC[i] değerleri*

3. KOMPAKT AES DİZAYNI ve STANDART AES DİZAYNI KARŞILAŞTIRMALI ANLATIM

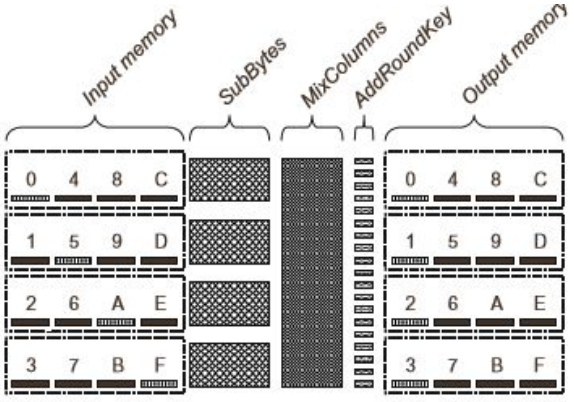
Bölüm 2'de anlatılanlar, standart bir AES encryption ve decryption işlem basamaklarını göstermektedir. Bu bölümde ise, standart algoritma basamaklarının kompakt bir FPGA dizaynı için nasıl uyarlandıklarına değinilmektedir.

3.1 Folded Mimari(Katmanlı mimari)

Şifreleme ve deşifreleme algoritmalarındaki round'lar incelendiğinde, Shiftrow ve Subbyte işlemlerinin tam ters sıralamada olduğu görülmektedir. Ancak Subbyte ve Shiftrow adımlarının sıralamasının değiştirilmesi, çıktılarda herhangi bir farklılığa sebep olmaz. Bunun sebebi, her iki işlemin byte'ları bozmadan işleme alması ve bu byte işlemlerinin de birbirlerinden bağımsız olması ve birbirlerini etkilememesidir. Yani byte'ları yer değiştirip Subbyte işlemine almış olmakla, aynı byte'ı Subbyte işlemine aldıktan sonra yer değiştirmenin arasında bir fark yoktur. Şekil 3'teki işlemler dikkatle incelendiğinde bu ayrıntı fark edilebilmektedir.

Yine Şekil 3'teki bu dört işlem dikkatlice incelendiğinde; 128-bit'lik girdinin sadece 32 bitlik bir kısmının diğer bitlere işlem uygulanmasına ihtiyaç duyulmadan şifrelenebilmesinin ve deşifrelenebilmesinin mümkün olduğu görülmektedir. Dolayısıyla her işlemin 128-bit'in tamamına uygulanmasını beklemeden; bir tane 32 biti 4 işlemden arka arkaya geçirebilmek teorik olarak olasıdır. Bu durumu fark eden akademisyen ekip, katmanlı bir yapı oluşturarak, 128 biti 32 bit halinle daha küçük parçalarla işleme sokmayı hedeflemiştir.

Gözlemlerini daha da ileriye taşıyan ekip, 128 bitin içerisinden doğru 32-biti seçtikleri takdirde Shiftrow işlemini otomatik olarak yapabildiklerini görmüşlerdir. Tüm bu çıkarımları bir araya getirdiklerinde ise 32 biti tek seferde şifreleyebilen ya da deşifreleyebilen katmanlı(folded) bir yapı oluşturmayı başarmışlardır.(Şekil 12).



Şekil 12. *AES 128-bit Folded(Katmanlı) Mimari*

Oluşturulan bu katmanlı yapıdan elde edilen sonuç şudur;

- 0, 5, A, F byte'ları okunur. Bu byte'lar üzerine Subbytes, Mixcolumns ve AddRoundkey işlemleri uygulanır. Çıkan sonuçlar çıktı hafızasına 0, 1, 2, 3 konumlarında yazılır. Böylece shift row işlemi de gerçekleştirilmiş olur.
- Yukarıdaki işlemler 4, 9, E, 3 girdileri için tekrarlanır ve sonuçları 4, 5, 6, 7 çıktı konumlarına yazılır.
- Yukarıdaki işlemler 8, D, 2, 7 girdileri için tekrarlanır ve sonuçları 8, 9, A, B çıktı konumlarına yazılır.
- Yukarıdaki işlemler C, 1, 6, B girdileri için tekrarlanır ve sonuçları C, D, E, F çıktı konumlarına yazılır.

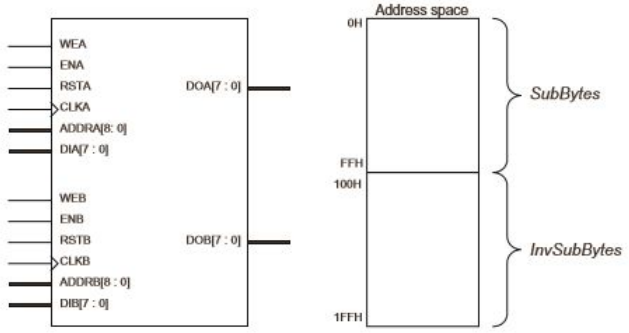
Yukarıda bahsedilen adımlar, 4 clock cycle içerisinde tamamlanmış olur. Böylece sadece 4 clock cycle'da, her defasında bir adet 32-bit işleme girerek, tüm 128 bitlik veri işleme girmiş olur. Bu bir round'luk işlemin tamamlanması demektir. Tüm bu işlemler 10 kez yapıldığında(en başta da bir Addroundkey işlemi olduğu düşünülürse) toplamda 44 clock cycle süre içerisinde 128 bitlik girdi için tüm şifreleme ya da deşifreleme işlemi bitmiş olur.

3.2 Folded Mimaride Addroundkey

Addroundkey işlem basamağı yalnızca "xor" işlemi içerdiğinden hem şifrelemede hem de deşifrelemede olduğu gibi kullanılır.

3.3 Folded Mimaride Subbyte ve InvSubbyte

Tablo 1 ve 2'de görüldüğü Subbyte ve InvSubbyte sonuçlarını bir tablo halinde tutabilmek mümkün. Bu makalede FPGA üzerinde tablolardaki değerleri tutabilecek iki farklı hafıza elemanına dikkat çekiliyor; LUT ve Block Ram. Üretilen alt anahtarları LUT'ta tutmaya karar verdikleri için Subbyte ve InvSubbyte elemanlarını RAM'de tutarak FPGA kaynaklarını dengeli bir şekilde kullanmanın daha mantıklı olduğunu düşünüyorlar. Şekil 13'te blok Ram için yapılan implementasyonu yer almaktadır.



Şekil 13. *Subbyte ve InvSubbyte için Block Ram İmplementasyonu*

3.4 Folded Mimaride Mixcolumn ve InvMixcolumn

AES 128 bit şifrelemede uygulanan Mixcolumn işlemi 2. Bölümde Şekil 7 ve 8'de ifade edilmiştir. Folded mimaride de bu işlemin Mixcolumn için aynen yapılması gerekir. 32 bitlik 1x4 girdi matrisi ile Şekil 8'deki 4x4 sabit değerlerin olduğu matris çarpılır. Bu matris c(x) fonksiyonu olarak ifade edilir(Denklem 1).

$$c(x)=\{03\}x3+\{01\}x2+\{01\}x+\{02\} \quad (1)$$

$$d(x)=c-1(x)=\{0b\}x3+\{0d\}x2+\{09\}x+\{0e\} \quad (2)$$

Deşifreleme işlemi için gerekli olan ise tersi olan fonksiyon $c^{-1}(x)$ yani d(x) fonksiyonudur(Denklem 2).

Makalede c(x) ve d(x) fonksiyonlarını aynı veri yolu üzerinde, mümkün olan en kompakt şekilde bir araya getirebilmek için, bir fonksiyonun kendisiyle çarpımının 1 olması gerektiği üzerinden yola çıkılmıştır(Denklem3). Eğer bir fonksiyon, kendisinin tersiyle iki kere çarpılırsa, bu fonksiyonun tersini elde edilir (Denklem 4).

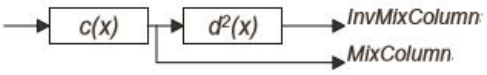
$$c(x)\bullet d(x)=\{01\} \quad (3)$$

$$c(x)\bullet d^2(x)=d(x) \quad (4)$$

Bu denklemler işleme girip düzenlendiğinde Denklem 5 elde edilmektedir. Burada dikkat edilmesi gereken şey; Denklem 5'te iki denklem elemanı artık denklemde yer almamaktadır. Bu da, bu fonksiyonun işlem kabiliyeti açısından avantaj sağlaması demektir.

$$d^2(x)=\{04\}x^2+\{05\} \quad (5)$$

Tüm bu bilgiler ışığında Mixcolumn yani c(x) fonksiyonunu ve InvMixcolumn yani $d^2(x)$ fonksiyonunu bir veriyolu üzerine ardışık biçimde yerleştirmek mümkün(Şekil 14).

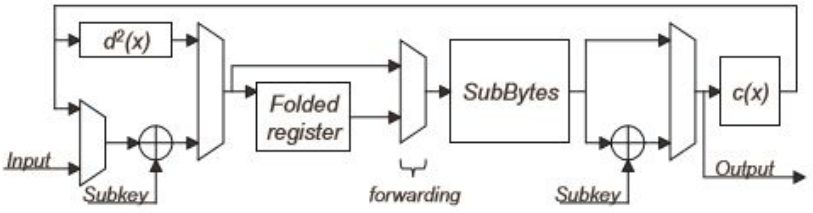


Şekil 14. *Mixcolumn ve InvMixcolumn İmplementasyonu*

3.5 Şifreleme ve Deşifreleme Ünitesi

Şimdiye kadar açıklanan şifreleme, deşifreleme algoritmaları ve bu algoritmaların basamaklarının FPGA üzerinde nasıl dizayn edilebilecekleri ile ilgili yaklaşımlar birleştirildiğinde Şekil 15'teki dizayn elde edilmektedir.

Bu dizaynda input(girdi) girişinden, 128 bitlik metin 4 seferde 32 bit olarak sisteme girecektir. Eğer sistem şifreleme yapmaktaysa, veri yolunda ilk subkey yazan "xor" yani Addroundkey işlemi uygulanacaktır. Folded register'da ise daha önce Şekil 12'de bahsedilen folded yapı yer alacaktır ve devamında yalnızca c(x) fonksiyonu uygulanacaktır. Eğer sistem deşifreleme yapıyorsa, İkinic subkeyin olduğu "xor" yani Addroundkey kullanılacaktır ve c(x) fonksiyonunun hemen ardından $d^2(x)$ fonksiyonunun olduğu veri yolu seçilecektir.

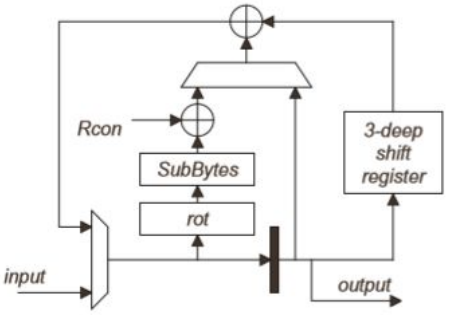


Şekil 15. *Şifreleme ve Deşifreleme Ünitesi*

3.3 Alt Anahtar Üretimi

Dizayn edilen folded mimaride 128 bitin tamamı yerine 32 bitlik parçalar işlemlerden geçmektedir. Bu da 32-bit girdi metin ile 32-bit alt anahtar parçasının işleme girmesini gerektirmektedir. Şekil 4'te görüldüğü gibi, alt anahtar üretilirken sırayla işleme girecek 32 bitlik parçalar halinde üretilmektedirler. Bu çalışmada, 32 bitlik alt anahtar üreten donanım dizaynının çıktıları bir Ram'e kaydedilmektedir.

Şifreleme işlemi sırasında alt anahtarlar paralel bir şekilde üretilebilir(Şekil 2), bunda herhangi bir problem yoktur ve hatta bu durum zamandan kazanç sağlar. Ancak bu makalenin konusu olan kompakt AES dizaynında hem şifreleme hem de deşifreleme işlemi yapılabilmelidir. Ve Şekil 2'de görüldüğü gibi, deşifreleme işleminde anahtarlar en son üretilen alt anahtardan kullanılmaya başlanır. Bu da deşifreleme işleminden önce tüm alt anahtarların bir kez üretilmiş olmasını zorunlu kılar. Dolayısıyla bu makalede, şifreleme ve deşifreleme işlemleri başlamadan hemen önce tüm alt anahtarlar, kullanılacakları 32 bitlik parçalar halinde üretilip Ram'e kaydedilmektedir. Daha sonra alt anahtar üretiminin bittiğini işaret eden bir sinyal sayesinde şifreleme ya da deşifreleme işlemine başlanır. Alt anahtarı üreten 32 bit girdi ve 32 bitlik çıktıya sahip olan component Şekil 16'daki gibi dizayn edilmiştir.



Şekil 16. *AES 32-bit alt anahtar üretmek için geliştirilmiş dizayn*

Daha önce de bahsedildiği üzere 11 kere 128 bitlik alt anahtar kullanılmaktadır. Her 128 bitlik alt anahtar için Şekil 6'da yer alan 32 bitlik dizaynın 4 kere koşması gerekmektedir. Bu nedenle tüm anahtarların üretimi 44 saat çevrimi(clock cycle) sürmektedir.

Her saat çevriminde mutlaka 32 bitlik çıktı(output) vermiş olmalıdır. Bu çıktı Ram'e kaydedilmelidir.

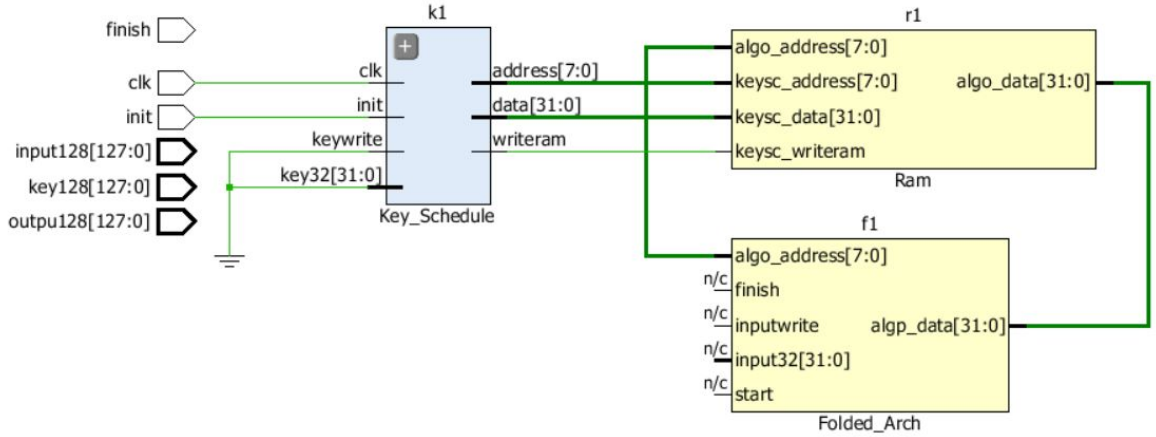
İlk 4 saat çevriminde, dizayna girdi(input) kısmından anahtarın 32 bitlik parçaları gelmelidir. Bu anahtar parçalarının ilk 3 tanesi doğrudan 3 deep shift register componenet'ine kaydolmalıdır ve başka işlem görmemelidir. Dördüncüsü ise hem 3 deep shift register componenet'ine kaydolmalı hem de rob, Subbytes ve rcon işlemlerinin yapıldığı path'ten geçerek 3 deep shift register'a ilk kaydedilen 32 bitlik input ile "xor" işlemine girmelidir. Böyle beşinci(5.) 32 bitlik round'umuz için alt anahtarı elde edilmiş olunur.

Bu noktadan sonra(4'ten büyük tüm roundlar) input olarak gösterilen bölümden herhangi bir girdi alınmamalıdır. Devre içinde üretilen alt anahtarlar hem output olarak Ram'e kaydedilmeli, hem 3 deep shift register'a kaydedilmelidir. Ancak 4 ve katları olan roundlar da mutlu Subbyte işleminin olduğu path tercih edilmelidir. Diğer tüm roundlarda ise doğrudan 3 deep shift register'ın çıktısı ile "xor" işlemi olan path takip edilmelidir. Ve çıktı yine devreyi besleyecek şekilde tekrar işleme sokulur. Tüm bu basamaklar 44 saat çevrimi boyunca tekrarlanmalıdır.

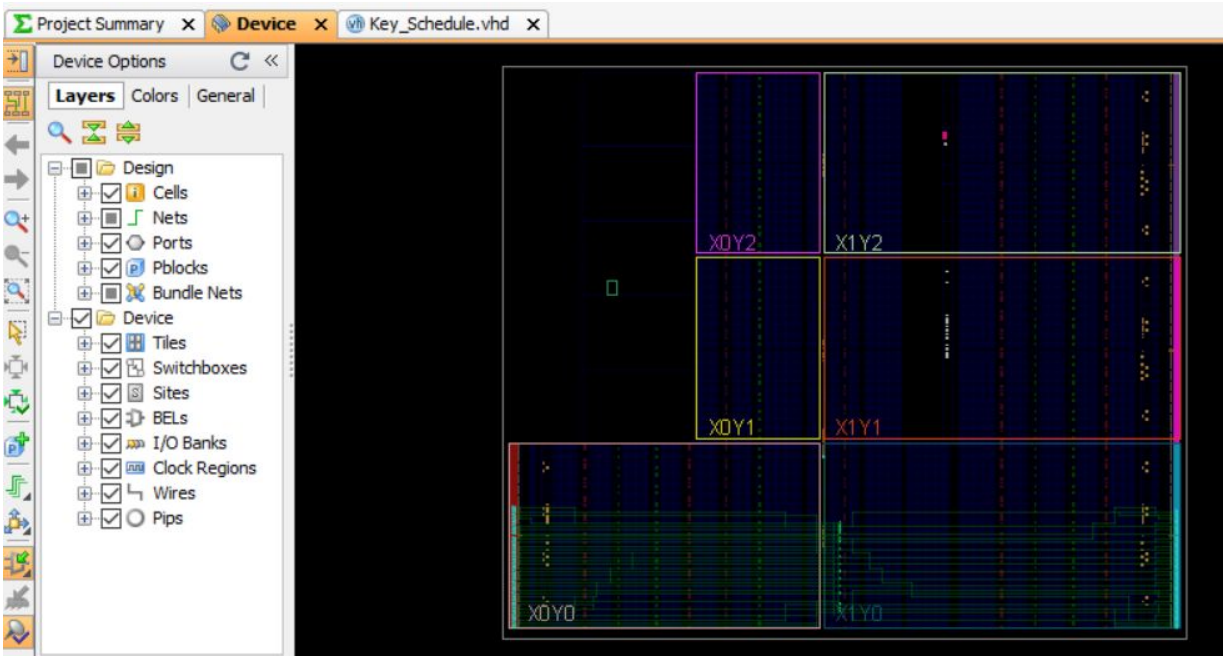
Alt anahtarların sırayla üretilmesi ve hem şifrelemede hem de deşifrelemede sırayla kullanılmasından dolayı FPGA hafızasında sıralı işlem için shift register olarak kullanılabilecek LUT elemanlarının kullanımı hedeflenmiştir.

4. SENTEZ, SİMÜLASYON, İMPLEMENTASYON

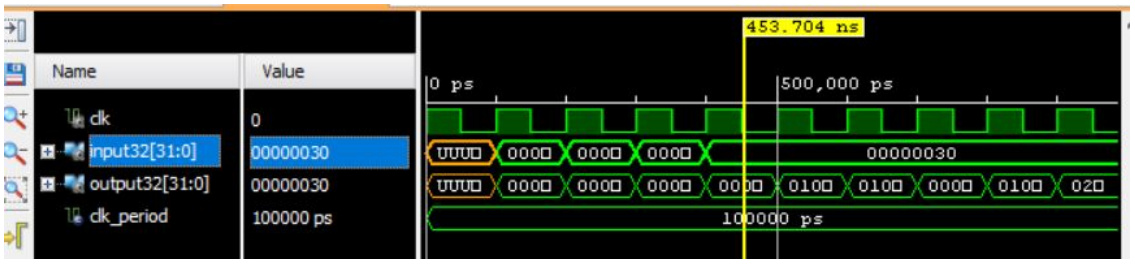
Şekil 16'da Key Schedule tüm işlem başlamadan önce 44 clock cycle dönecek, daha sonra üretilen değerler Ram e kaydedilecek ve Folded_Arch component'inde implement edilen şifreleme/deşifreleme ünitesi Ram'deki keyleri kullanmaya başlayacak. Şifreleme ya da deşifreleme işlemini de yine 44 clock cycle'da bitirecek.



Şekil 17. *VHDL Top Dizayn*

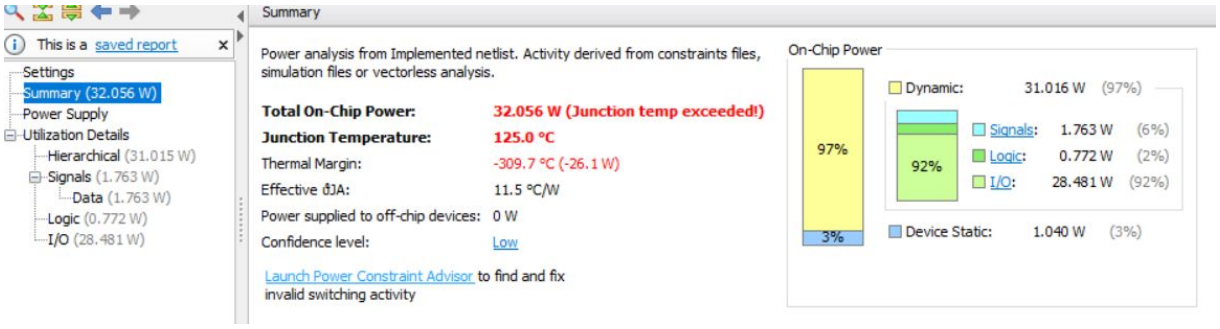


Şekil 18. *VHDL Implemented Design*

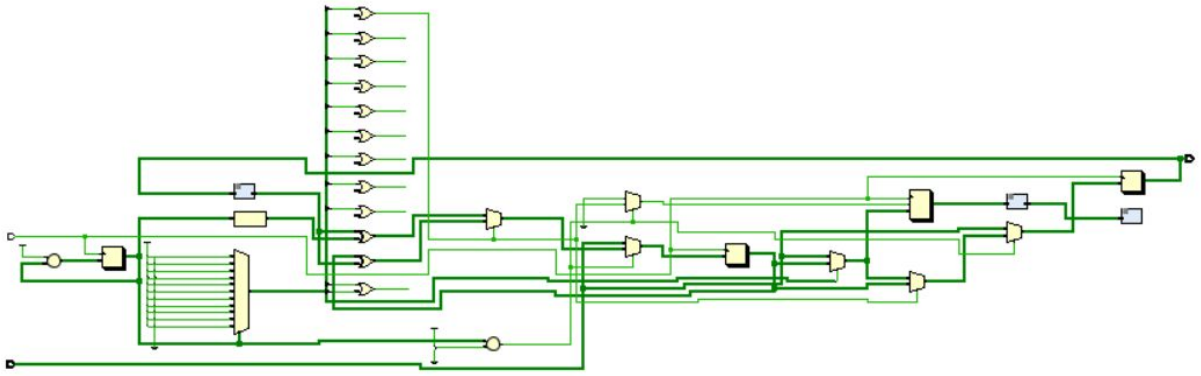


Şekil 19. *Alt Anahtar Üretim Simülasyonu*

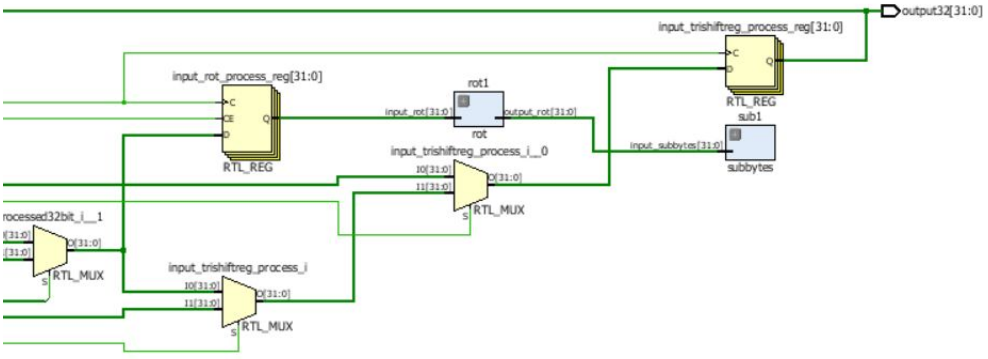
Şekil 19'da ilk 4 input direkt olarak output'a gitmekte. Dördüncü inputla beraber, alt anahtarlar devre içerisinde bir önceki alt anahtardan üretilmeye başlanıyor.



Şekil 20. *Özet*



Şekil 21. *Alt Anahtar Üretimi RTL Şeması*



Şekil 22. *Alt Anahtar Üretimi RTL Şeması - 2*

5. SONUÇLAR

Kompakt AES algoritması implementasyonu sayesinde;

- Mixcolumn ve InvMixcolumn işlemlerine farklı bir yaklaşım getirilmiştir.
- Aynı dizayn içerisinde hem şifreleme hem de deşifreleme işlemleri yapılabilmektedir.
- 128 bitin tamamı işleme alınmaya çalışılmadığı için ara değerleri tutma konusunda çok daha az yer kaplanmıştır.
- Daha az devre elemanı harcandığı için maliyet bakımından daha düşüktür.

5. KAYNAKÇA

Pawel Chodowiec and Kris Gaj, *Very Compact FPGA Implementation of the AES Algorithm*, USA pp.319-333

Christof Paar and Jan Pelzl, Understanding Cryptography, *The Advanced Encryption Standard (AES)*87-120.