

HDD(Hardware Description Document)

HW based AES

차세대반도체학과

우상욱(dndn3515@ssu.ac.kr)

Revision History

Date	Version	Description	Author
2024/06/20	V0.0	HDD template	Inchul.Song
2024/06/28	V0.1	Intial HDD	Inchul.Song
2025/01/07	V0.2	Overall draft	SangWook.Woo
2025/01/08	V0.3	Explain Sub Block and timing diagram	SangWook.Woo
2025/01/14	V0.4	Modified timing diagram and Explain text	SangWook.Woo
2025/01/15	V0.5	Modified overall Block diagram and Explain text	SangWook.Woo
2025/01/15	V1.0	All completed	

Table of Contents

1.1. Overview.....	5
1.1.1. Description	5
1.1.2. Feature.....	7
1.2. Overall block	8
1.2.1. Block diagram	8
1.2.2. IO description.....	9
1.2.3. Memory list.....	9
1.2.4. Register map	10
1.3. Sub block description	11
1.3.1. Sub block #1.....	12
1.3.2. Sub block #2.....	13
1.3.3. Sub block #3.....	14
1.3.4. Sub block #4.....	16
1.4. Further work.....	16
1.5. Appendix	16

1.1. Overview

Multimode Modem 을 지원하는 smart phone 에서 CP(Communication Processor)와 AP(Application Processor) 사이에 주고받는 data 는 암호화 되어있으며, 3G 는 Kasumi, Snow3G 암호화 알고리즘을, 4G & 5G 는 ZUC, AES 알고리즘을 사용하고 있다. Smart phone 에서 3G 까지는 SW based 로 암호화 기능을 수행했지만, 4G 를 지나 5G 에서 data rate 이 증가하여 기존 SW based 암호화 처리로는 원하는 성능을 만족할 수 없게 되었다. 이에 수행되는 모든 암호화 알고리즘을 HW 로 구현하였고 이를 통해 4G, 5G data rate 을 만족하게 되었다. 본 문서에서는 SW based AES 가 아닌 5G 까지의 performance requirement 를 만족하도록 설계한 HW based AES(AES-128 @mobile) 구조 및 동작을 설명하였다.

1.1.1. Description

기존 SW based 로 구현되었던 AES 의 경우 5G(~14Gbps)에서 요구되는 성능을 만족시키지 못하기 때문에 HW based AES 가 필요하다.

AES 는 128bit, 192bit, 256bit 에따라 반복되는 round 의 수가 다르다. 설계하고자 하는 목표는 AES128 을 기준으로 하여 10 번의 round 가 필요하다. 각 round 는 AddRoundKey, SubBytes, ShiftRows, MixColumns 단계로 구성되어 있다.

AddRoundKey 단계에서는 각 round 별로 생성된 key 값을 plainText 에 XOR 연산을 진행한다. SubBytes 단계에서는 보통 S-Box 라고 불리는 약속되어있는 Table 에 값을 Mapping 하여 값을 치환한다. 이 과정으로 비선형성을 확보한다. (Figure1.1.1 참조)

ShiftRows 단계는 4X4 state 에 각 row 의 번호만큼 shift 시킨다. row0 번은 shift 하지 않고 row1 는 한칸씩 row2 는 두칸씩 row3 는 세칸씩 이동시킨다.

MixColumns 단계에서는 열을 섞어주는 단계이다. 이 단계에서는 행렬곱 연산을 이용하여 기능을 구현하게 된다. (Figure1.1.2 참조)

전체적인 Flow 는 Figure1.1.3 과 같다. Encryption 과정에서 AddRoundKey 를 한 번 수행한 후 n 번 round 를 수행한다. 마지막 n 번째 단계에서는 Mixcolumns 를 제외한 SubByte, ShiftRows, AddRoundKey 단계를 수행하면 ciphered text 가 나오게 된다.

	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

Figure1.1.1 S-Box Table

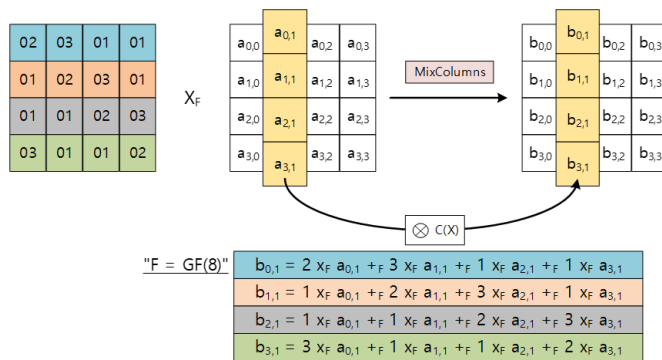


Figure1.1.2 MixColumn 행렬곱

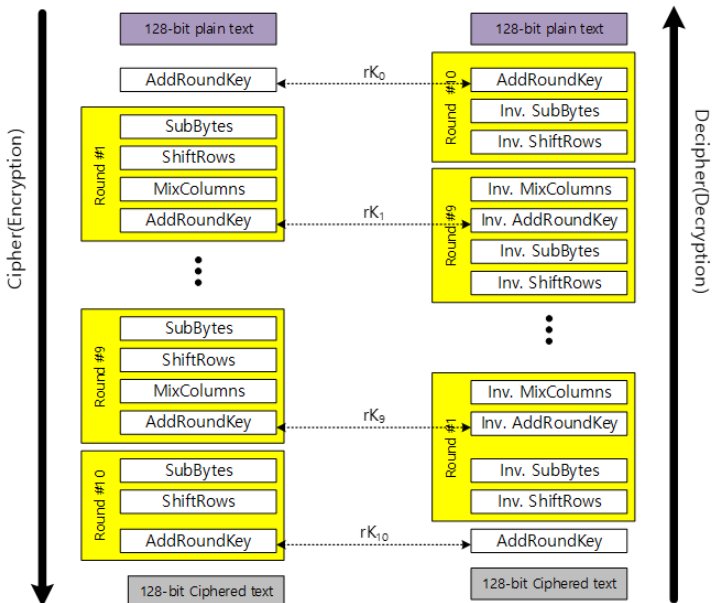


Figure1.1.3 AES 순서

1.1.2.Feature

이번 인턴과정에서 구현한 AES 는 APB Interface 를 사용하여 32 비트 단위로 받은 data 4 개를 모아서 128bit 로 inbuf.v module 에 저장한다. 이때 APB Interface signal 중 wait 와 slave error 와 관련된 PREADY, PSLVERR 은 사용하지 않는다.

AES 하드웨어는 big endian 을 기준으로 데이터를 암호화하기 때문에 software 에서 받은 little endian data 를 변화하는 과정을 거친다. 다만 모든 data 는 비어있는 bit 없이 128 비트의 배수만큼 크기가 온다고 가정하므로 zero padding 을 생략한다.

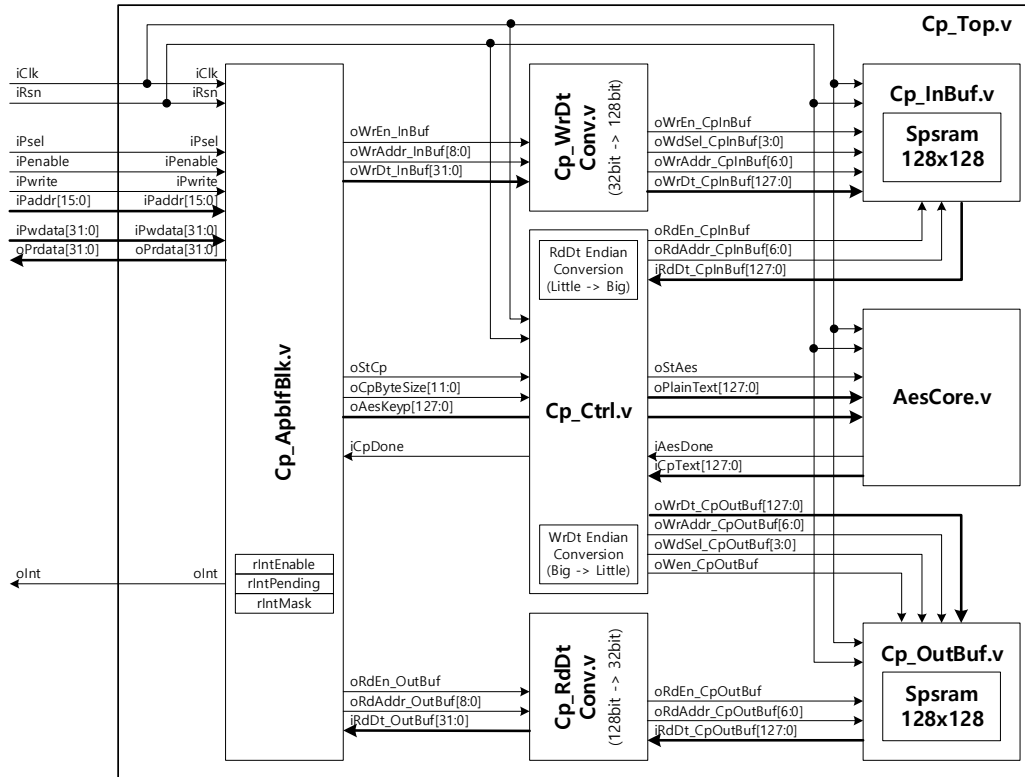
InBuf 에 plaintext 를 저장하고 AES 암호화를 하여 OutBuf 에 Cyphered text 를 저장한다. 이 모든 과정은 control block 에서 제어한다.

AES-128 은 한 번의 pre round, 모든 단계를 실행하는 round 가 9 번 반복되고, 3 개의 단계만 실행하는 round 가 한 번 실행된다. 그러므로 round 기능을 한 개 만들어서 각 단계를 enable 신호로 제어한다. 이번 HW based AES 의 구현은 암호화 과정만 구현한다. 추후에 복호화를 구현하여 direction 변수를 사용하여 암호화 및 복호화가 모두 가능한 하드웨어를 설계할 예정이다.

1.2. Overall block

1.2.1. Block diagram

설계한 전체 Block diagram



APB Bus 에서 들어오는 데이터는 32 비트이다. 128 비트의 데이터를 받으려면 4 번의 read 가 필요하다. 1 번 read 할 때 32 비트 데이터를 128 비트로 확장시킨다. 만들어진 128 비트를 enable 신호로 제어하여 원하는 자리수의 비트를 원하는 주소값에 write 한다.

AES core 에서는 InBuf 에서 128 비트씩 읽어와서 암호화한다. 각 round 별로 12clk 이 소요되며 128bit cyphered text 는 OutBuf 에 저장된다.

OutBuf 에 초기 byteSize 만큼 암호화가 완료되어 저장되었으면 APB interface 를 통해 interrupt 를 발생시켜 암호화가 완료되었음을 알린다.

APB interface 로 암호화된 데이터를 읽어올 시에는 32 비트씩 읽어오며 endian 이 다르기 때문에 conversion 과정을 거쳐서 출력된다.

1.2.2.IO description

Table 1 Input

Name	Bit width	Initial value	Source Block	Description
iClk	1	0	CMU	300MHz clock input
iRsn	1	1	CMU	Active low reset
iPsel	1	0	Bus Matrix	Select signal for slave
iPenable	1	0	Bus Matrix	Enable signal for slave
iPwrite	1	0	Bus Matrix	Write enable (0/1 : r/w)
iPaddr	16	16'h0	Bus Matrix	APB Bus Address
iPwdata	32	32'h0	Bus matrix	Write data

Table 2 Output

Name	Bit width	Initial value	Destination Block	Description
oPRDATA	32	N/A	AXI to APB Bridge	APB read data
oInt	1	0	Bus Matrix	APB Interrupt data

1.2.3.Memory list

Table 3 Memory list

Name	Location	Type	Size	Description
InBuf	AesTop.A_InBuf	SP SRAM	128*128	Max packet size: 2kBytes data bit width : 1282bit, so 128depth
OutBuf	AesTop.A_OutBuf	SP SRAM	128*128	Max packet size: 2kBytes data bit width : 1282bit, so 128depth

APB 의 bit width 는 32bit 이지만 AES-128 방식을 사용하므로 구현의 편의성을 위해 buffer 에 들어가는 SpSram 의 bit width 를 128bit 로 하였다. 총 4kBytes 가 사용된다.

1.2.4. Register map

	NAME	Offset	Range	Field	Reset	Access	Description
B	AES	0x7000_0000	0xFFFF				
S	Common						
R	StartCmd	0x0000	[0:0]		0x0	W	NatHw start command
F			[0:0]	StartCmd	0x0		Set to 1 by SW: NatHw start
R	DataWordSize	0x0004	[31:0]		0x0	R/W	Input data word size (1 word = 4 Byte, 1~512 value)
F			[31:0]	DataWdSize	0x0		
S	Control						
R	AesKey_31_0	0x2000	[31:0]		0x0	R/W	128bit AES key value
F			[31:0]	AesKey_31_0	0x0		AES key[31:0]
R	AesKey_63_32	0x2004	[31:0]		0x0	R/W	128bit AES key value
F			[31:0]	AesKey_63_32	0x0		AES key[63:32]
R	AesKey_95_64	0x2008	[31:0]		0x0	R/W	128bit AES key value
F			[31:0]	AesKey_95_64	0x0		AES key[95:64]
R	AesKey_127_96	0x200C	[31:0]		0x0	R/W	128bit AES key value
F			[31:0]	AesKey_127_96	0x0		AES key[127:96]
R	AesDirection	0x2010	[0:0]		0x0	R/W	AES-direction
F			[0:0]	AesDirection	0x0		1'b1-Encryption, 1'b0-Decryption
S	InBuf						
R	InBufAddr_0	0x4000	[31:0]		0x0	W	AES InBuf (2kB : 512depth * 4Bytes)
F			[31:0]	InBufAddr_0	0x0		InBuf address 0 data write
R	InBufAddr_1	0x4004	[31:0]		0x0	W	
F			[31:0]	InBufAddr_1	0x0		InBuf address 1 data write
R	[31:0]		0x0	W	...
F			[31:0]	...	0x0		...
R	InBufAddr_511	0x47FC	[31:0]		0x0	W	
F			[31:0]	...	0x0		InBuf address 511 data write
S	OutBuf						
R	OutBuf1_Addr_0	0x6000	[31:0]		0x0	R	AES OutBuf #1 (2kB : 512depth * 4Bytes)
F			[31:0]	OutBuf1_Addr_0	0x0		OutBuf #1 address 0 data read
R	OutBuf1_Addr_1	0x6004	[31:0]		0x0	R	
F			[31:0]	OutBuf1_Addr_1	0x0		OutBuf #1 address 1 data read
R	[31:0]		0x0	R	...
F			[31:0]	...	0x0		...
R	OutBuf1_Addr_511	0x67FC	[31:0]		0x0	R	
F			[31:0]	OutBuf1_Addr_511	0x0		OutBuf #1 address 511 data read
S	Int						
R	IntEnable	0xA000	[0:0]		0x0	R/W	AES interrupt enable register
F			[0]	IntEnable	0x0		Set to 1 by SW: AES interrupt pending register Enable
R	IntPending	0xA004	[0:0]		0x0	R/W	AES interrupt pending register
F			[0]	IntPending	0x0		When interrupt enable is On and AES operation is done, HW write 1 to make interrupt. If SW reads 1 and then writes 1 to clear
R	IntMask	0xA008	[0:0]		0x0	R/W	AES interrupt mask register
F			[0]	IntMask	0x0		Set to 1 by SW: AES interrupt transfer to interrupt output

0x4000 번대의 data 는 inBuf 의 SpSram 에 저장되며 write 만 가능하고 0x6000 번지의 data 는 read 만 가능하며 OutBuf 에 SpSram 에 저장된다.

1.3. Sub block description

- APB interface

Psel, Penalbe, Pwrite 신호의 맞게 data 를 address 에 맞게 읽고 쓰는 역할을 한다.

- Register block

Register Map 에 맞게 디자인되어 데이터를 읽고 쓴다. 또한 내부 Controller 나 interrupt 제어를 위해서 APB interface 가 Register block 을 이용한다.

- Main Controller

APB 에서 데이터를 읽어와 InBuf 에 저장하고 완료되면 암호화를 시작하여 OutBuf 에 암호화된 정보를 저장하게 하는 흐름을 제어하는 역할이다. Endian conversion 을 수행하며 Buf, AesCore 를 제어하기 위해 다양한 enable 신호를 출력한다.

- AES Core block

AES-128 방식을 활용하여 InBuf에서 읽어온 값을 암호화 한다.

- AES control block

AES 내부에는 round 하나만 설계되고 control block 에서 round 내부의 4 가지 단계를 flag 신호로 제어한다.

- AES RoundFunction Block

AES 의 각 Round 와 Key Expansion 이 구현되어 있고 Control block 에서 발생한 flag 신호로 각 round 에 맞는 키와 연산을 수행한다.

- InBuf / OutBuf

외부 신호를 받아 SpSram 에 값을 읽고 쓸 수 있게 한다.

- WrDt Conversion/RdDt Conversion

WrDt Conversion 은 APB interface 의 32bit data 를 128bit data 로 Conversion 한다.

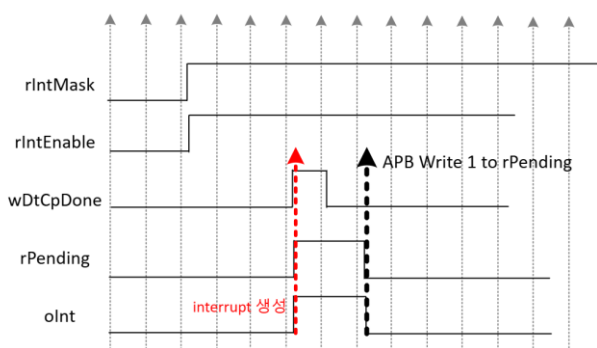
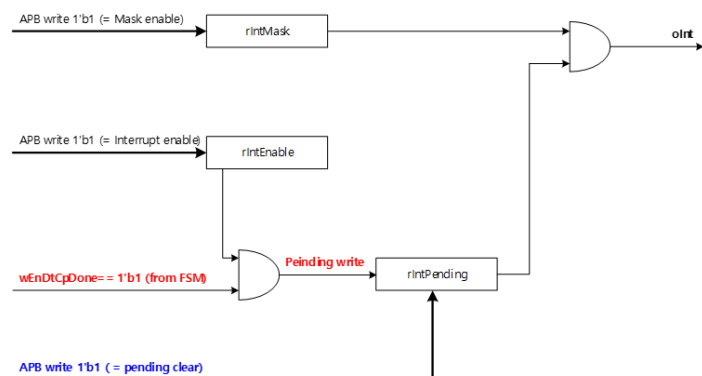
RdDt Conversion 은 SpSram 의 128bit data 를 32bit 씩 출력한다.

1.3.1.Sub block #1

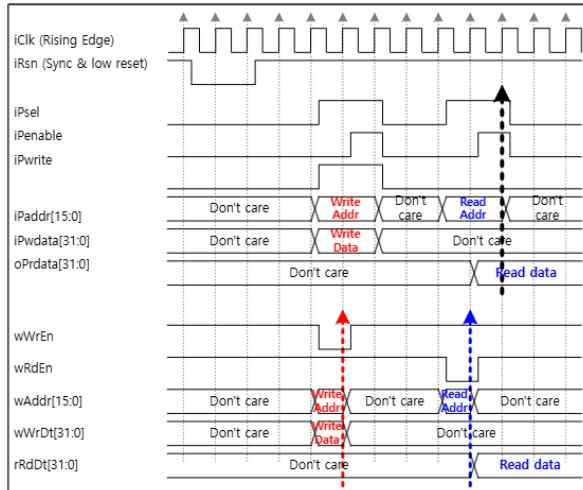
APB Block to Register block

register map의 조건에 맞게 read, write할 수 있게 한다. Cp_ctrl block에 맞게 interrupt를 발생시킨다.

Interrupt register 구성



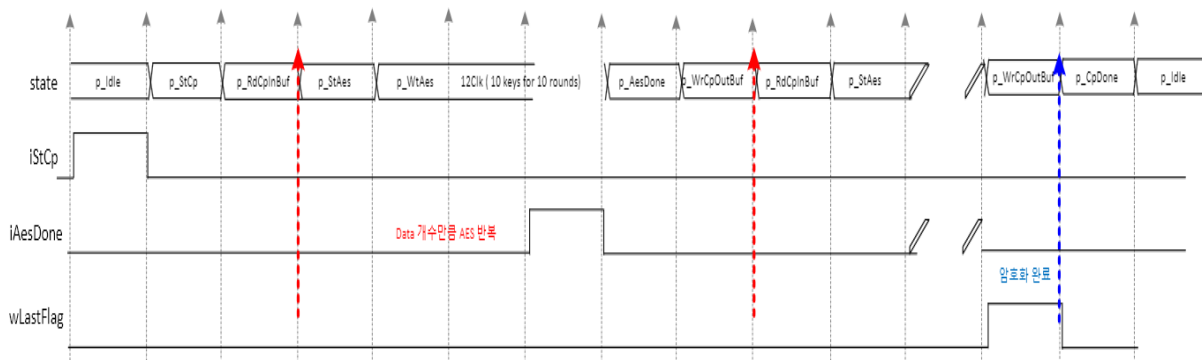
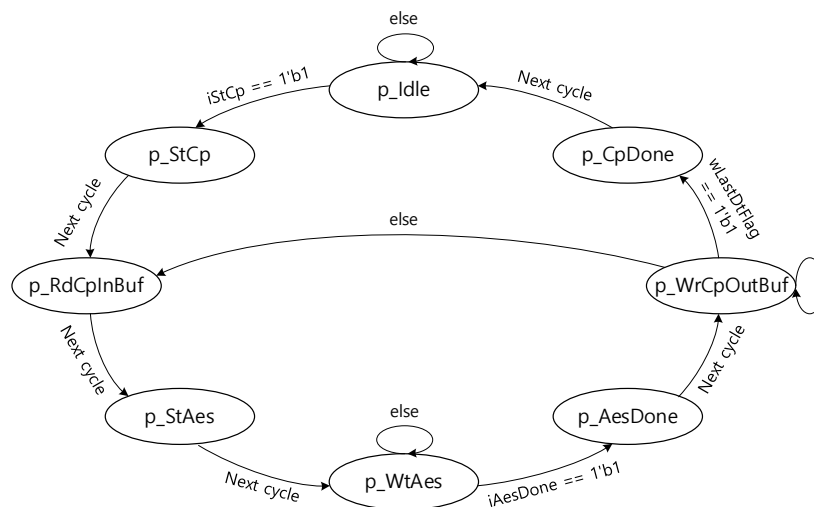
DtCpDont이 high가 되면 rPending값을 읽어 interrupt를 처리하고 rPending값에 1을 write하여 0으로 초기화한다.



구현한 APB interface 의 read, write timing diagram 은 다음과 같다.

1.3.2.Sub block #2

Main Controller

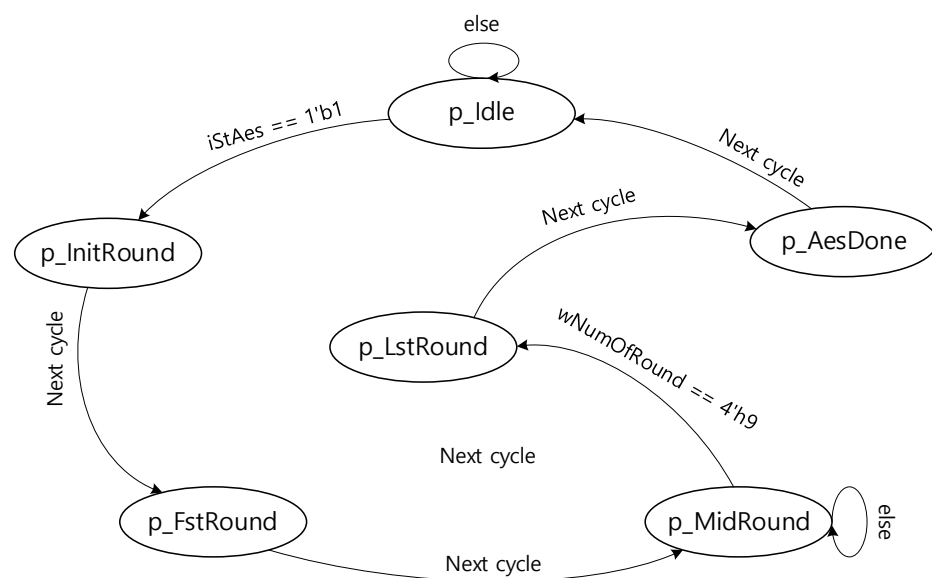
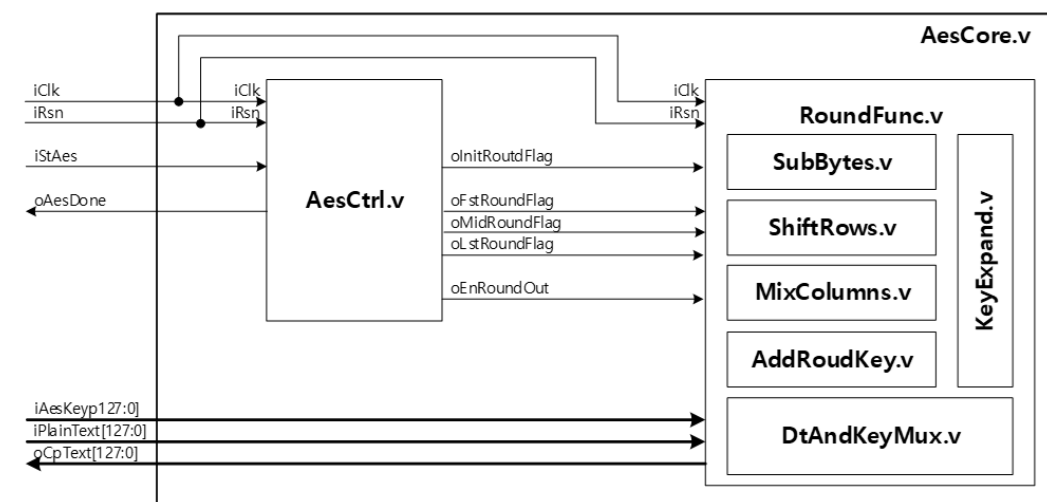


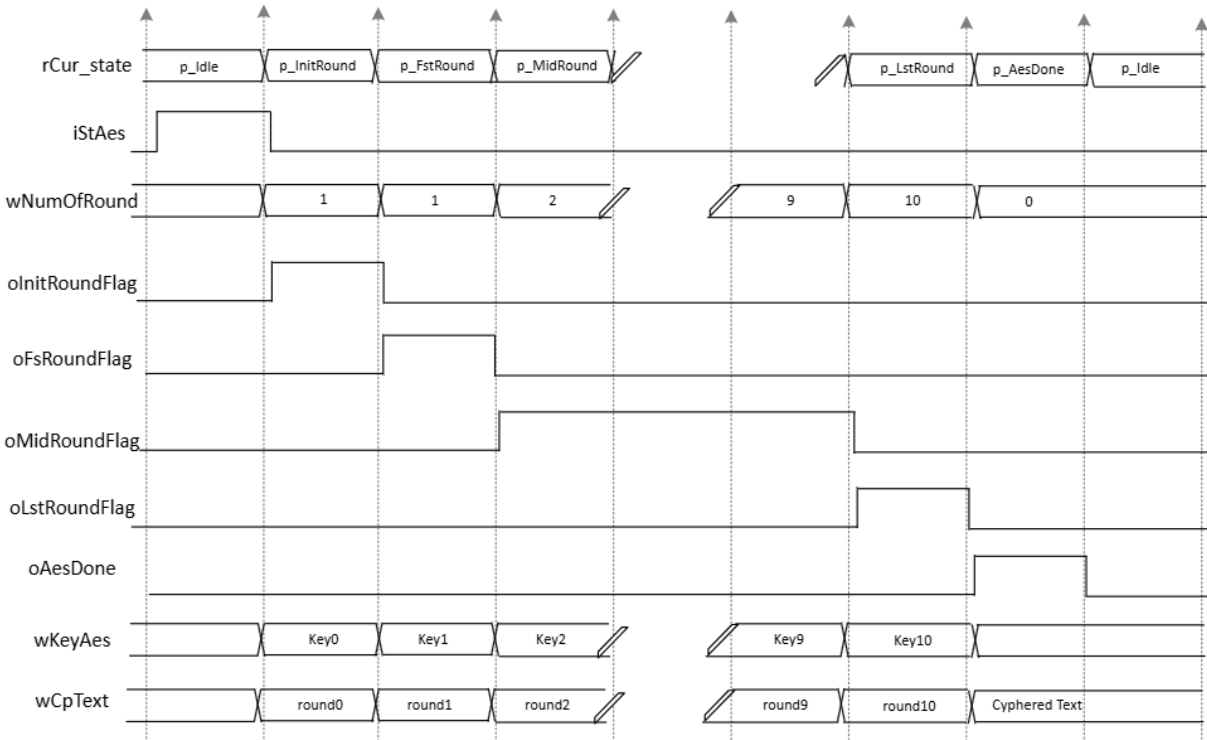
Main Controller 의 FSM 과 timing diagram 은 다음과 같다. iAesDone 는 전체 암호화가 끝날 때마다 1 이 되어 OutBuf 에 쓸 준비를 한다.

InBuf 에 모든 데이터가 암호화되어 OutBuf 에 쓰였다면 wLastDtFlag 를 1 로 출력하여 APB interface 에서 interrupt 를 발생시키게 한다.

1.3.3.Sub block #3

AES Core Block

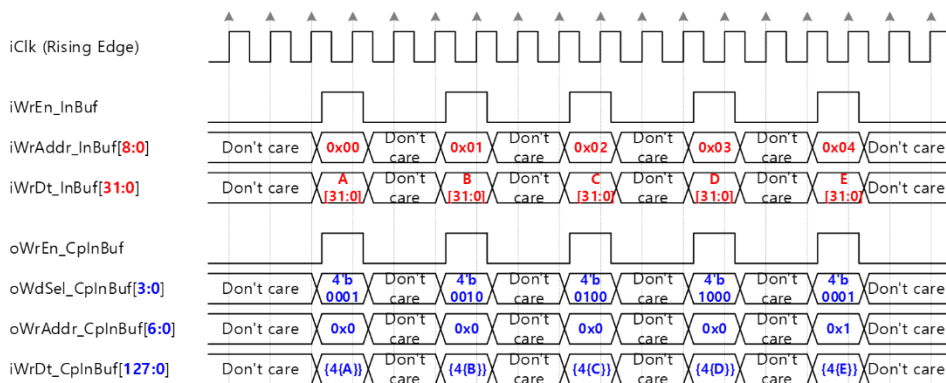




1.3.4. Sub block #4

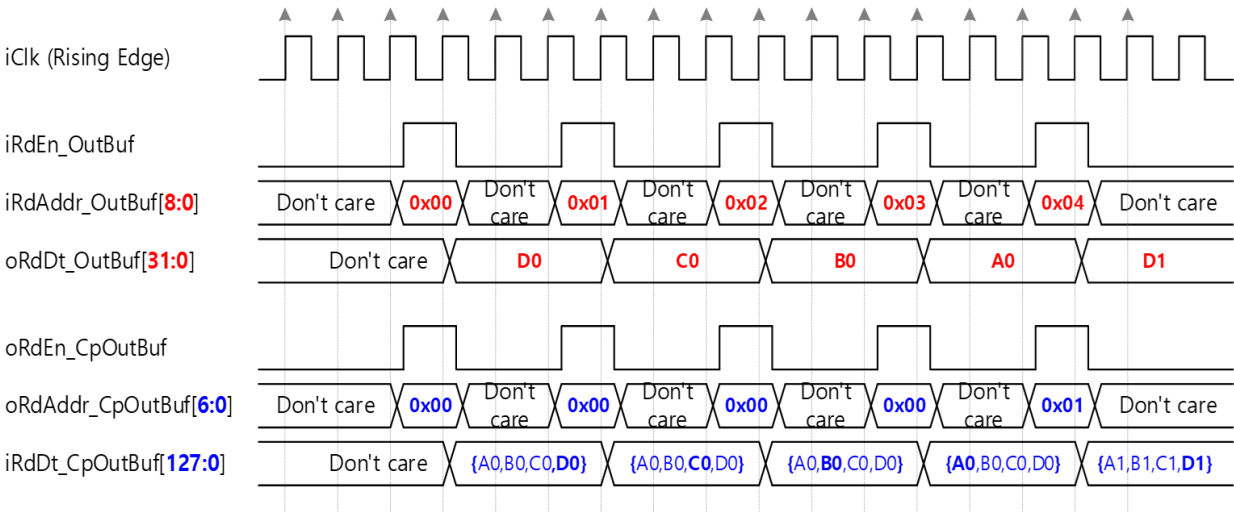
WrDtConv / RdDtConv

* Write function(4Bytes to 16Bytes conversion) @Cp_WrDtConv.v



위의 타이밍 다이어그램과 같이 32 비트 데이터 4 개를 받아서 Sram 에 저장한다. 이때 APB interface 의 주소값은 0x00, 0x04, 0x08, 0x0C 와 같이 4 씩 증가하므로 iWrAddr_InBuf 는 16 비트 주소값에 [10:2]만 사용한다. OWrAddr_CpInBuf 은 4 번 쓸때 1 씩 증가하므로 iWrAddr_InBuf 에 [8:2]를 전달해주면 된다.

* Read function(16Bytes to 4Bytes conversion) @Cp_RdDtConv.v



iRdAddr_OutBufd 의 [8:2]를 oRdAddr_CpOutBuf 에 전달하여 위 timing diagram 을 만족시킬 수 있다. iRdDt_CpOutBuf 의 값을 lsb 부터 32 비트씩 APB interface 의 read 신호에 맞게 출력해주게 된다.

1.3.5.Sub block #45

InBuf, OutBuf

InBuf, OutBuf 는 외부에서 들어온 신호를 SpSram 에 맞게 변환하여 올바른 동작을 할 수 있도록 한다.

1.4. Further work

Byte 수가 128bit 에 맞게 들어오지 않을 때 endian 에 맞는 zero padding 을 구현하지 않아 추후에 보완할 예정입니다.

또한 복호화를 구현하여 암호화 및 복호화를 모두 진행할 수 있도록 개선할 것입니다.

AES Block 을 한 개 사용하였지만 여러 개를 사용하여 clk 을 줄여 암호화의 속도를 개선할 것입니다.

power 의 효율성을 위해 dynamic Clock gating 기법을 추가할 예정입니다.

1.5. Appendix

<https://ko.wikipedia.org/wiki/%EC%97%94%EB%94%94%EC%96%B8>

<https://youtu.be/O4xNJsitN6E?si=8X8lbifwc8HgljFi>

<https://youtu.be/1X-R1var9-k?si=q1xuZsxMYLju4AS9>

<https://blog.naver.com/wnrjsxo/221711255389>

<https://github.com/michaelehab/AES-Verilog>