HDD(Hardware Description Document)/

HW based AES

차세대반도체학과

한 지 윤

Revision History

Date	Version	Description	Author
2024/06/20	V0.0	HDD template	Inchul.Song
2024/06/28	V0.1	Initial HDD	Inchul.Song
2025/01/07	V0.2	Initial Filling HDD	Jiyun_Han
2025/01/09	V1.0	Draft Complement	Jiyun_Han
2025/01/13	V1.1	First Revision	Jiyun_Han
2025/01/15	V1.2	Final	Jiyun_Han

Table of Contents

1.1. Overview

차세대 반도체 학과_학부생 연구 인턴 과정을 진행하면서 유무선 통신의 암호화 표준으로 사용되는 대칭형 알고리즘인 AES에 대해 수강하게 되었다. AES의 구조와 알고리즘과 En/Decryption 동작을 이해하고, SoC Interface를 갖도록 Verilog-HDL을 사용하여 128bit AES를 구현 및 검증해보려고 한다.

Multimode Modem을 지원하는 smart phone에서 CP(Communication Processor)와 AP(Application Processor) 사이에 주고받는 data는 암호화 되어있으며, 3G는 Kasumi, Snow3G 암호화 알고리즘을, 4G & 5G는 ZUC, AES 알고리즘을 사용하고 있다. Smart phone에서 3G 까지는 SW based로 암호화 기능을 수행했지만, 4G를 지나 5G에서 data rate이 증가하여 기존 SW based 암호화 처리로는 원하는 성능을 만족할 수 없게 되었다. 이에 수행되는 모든 암호화 알고리즘을 HW로 구현하였고 이를 통해 4G, 5G data rate을 만족하게 되었다. 본 문서에서는 SW based AES가 아닌 5G까지의 performance requirement를 만족하도록 설계한 HW based AES(AES-128 @mobile) 구조 및 동작을 설명하였다.

1.1.1. Description

AES(Advanced Encryption Stadard)는 데이터 암호화를 위해 사용되는 대칭형 블럭 암호화 알고리즘으로 높은 안전성(보안성)과 속도, 그리고 효율성으로 인해 인기를 얻어 전 세계적으로 많이 이용되고 있다. 기존 SW 기반 AES는 주로 CPU에서 직렬 처리로 실행되기 때문에 처리 속도가 제한적이며, 특히 5G같은 고속 통신 환경에서는 요구 성능을 충족하지 못한다. 이에 따라 HW기반 AES가 도입되어 병렬 처리와 최적화된 하드웨어 설계를 통해 높은 처리 속도와 에너지 효율성을 제공하기 때문에 HW based AES의 필요성이 강조되고 있다.

AES는 데이터를 블록 단위로 처리하며, 기본 기능은 암호화를 통해 암호문을 생성하고, 복호화를 통해 원본 데이터를 복원한다. 또한, 비선형 변환과 키 혼합을 통해 키를 추측하기 어렵게 보안성을 강화하며, SW와 HW구현 모두 높은 성능을 발휘하는 효율성을 가진다.

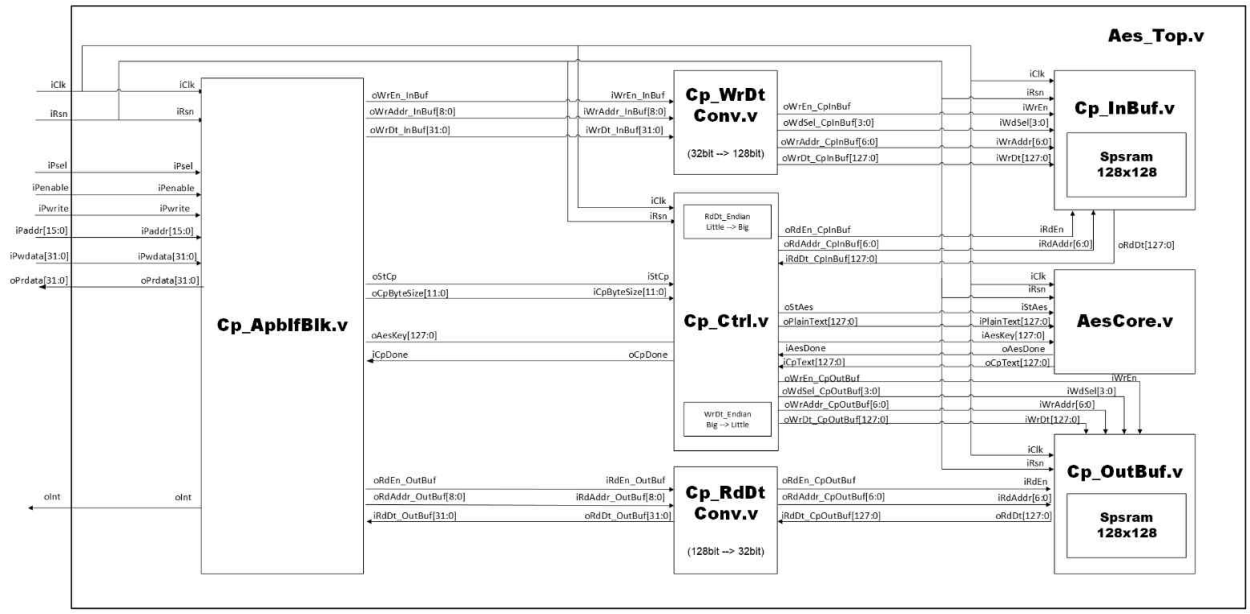
AES의 동작은 키를 확장하고, 데이터를 변환하는 과정을 여러번 반복하여 보안을 강화한다. 데이터를 변환하는 과정을 Round라고 하며 각 Round에서는 SubBytes, ShiftRows, MixColumns, AddRoundKey 과정을 반복하여 암호화한다. (단, 마지막 Round에서는 MixColumns이 빠진다.)

1.1.2. Feature

이번 인턴과정에서 구현한 AES는 32bit Data를 입력받아 128bit Memory에 저장하여 암호/복호화를 시키는 방식으로 AES-128을 설계했다.

1.2. Overall block

1.2.1. Block diagram



- Cp_ApbIfBlk 에서 Address와 Data를 받아 Write/Read동작 선언
  평문을 받아 Controller에 전송 및 암호키를 받아 AesCore에 전송
- Wr/RdDt Conv에서 16bit Address를 8bit로 새로 지정하여 원하는 Address의 Data를 Buffer에 Write/Read
- Ctrl에서 Start Signal을 받아 Data를 읽고 Endian Conversion하여 AesCore에 전달
- AesCore에서 암호화
- 다시 Ctrl에서 암호화 된 Data를 받아 저장 및 Apb로 전송
- 모든 작업이 완료되면 Interrupt 신호 전송

1.2.2. IO description

Table 1 Input

Name	Bit width	Initial value	Source Block	Description
iClk	1	0	CMU	MHz Clock Input
iRsn	1	1	CMU	Active Low Reset
iPsel	1	N/A	Bus	Active High Select
iPenable	1	N/A	Bus	Active Low Enable
iPwrite	16	N/A	Bus	Low = Read / High = Write
iPaddr	32	N/A	Bus	32bit Address
iPwdata	32	N/A	Bus	32bit 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	Interrupt Signal

1.2.3. Memory list

Table 3 Memory list

Name	Location	Type	Size	Description
InBuf	Cp_InBuf	SP SRAM	128*128	Max packet size: 2kBytes APB data bit width : 128bit, so 128depth
OutBuf	Cp_OutBuf	SP SRAM	128*128	Max packet size: 2kBytes APB data bit width : 128bit, so 128depth

1.2.4. Register map

	NAME	Offset	Range	Field	Reset	Access	Description
B	AES(Cipher only)	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	DataByteSize	0x0004	[31:0]		0x0	R/W	Input data byte size (1~2048 value)
F			[11:0]]	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: Encyption, 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

1.3. Sub block description

Cp_Top : 최상위 Top 모듈

- Cp_ApbIfBlk : Wr/RdDtConv로 주소/데이터를 전송, Ctrl의 FSM동작 신호를 전송 및 회신
  : 평문과 Key Data를 전송하고 암호문을 전달받아 출력

- Cp_Ctrl : InBuf에서 데이터를 받아 AesCore에서 변형된 데이터를 OutBuf로 이동

- AesCore : AES Round알고리즘으로 암호화

- Cp_WrDtConv : Apb에서 받은 데이터를 InBuf에 전송

- Cp_RdDtConv : OutBuf에서 받은 데이터를 Apb로 전송

- Cp_InBuf : Write된 데이터를 내부 Spsram에 저장

- Cp_OutBuf : 변환된 데이터를 내부 Spsram에 저장

1.3.1. Cp_ApbIfBlk

Description

iPsel, iPenable, iPwrite신호와 iPaddr로 들어오는 Address를 통해 Write/Read신호를 결정
--> Write신호일 때 16bit Address를 9bit Address로 변환하여 iPwdata를 WrDtConv로 전송 /
Read 신호일 때 RdDtConv에서 데이터를 받아 oPrdata로 출력

Address가 0x0000일 때 Cp_Ctrl가 작동되도록 설계, 0x0004일 때 Cycle횟수를 전송
데이처 처리가 끝나면 iCpDone신호를 받아 Interrupt신호를 전송

Address가 0xA0000일 때 IntEnable신호를 결정. iCpDone신호와 함께 Pending신호를 결정
0xA0004일 때 High Signal에서 Pending을 Clear
0xA0008일 때 IntMask 신호를 결정
Mask & Pending으로 oInt신호를 결정

Connect Interface

- Cp_Ctrl

- Address가 0x0000일 때 oStCp신호를 전송 --> Ctrl에서 FSM 수행
- Address 가 0x0004일 때 Cycle횟수를 전송
- Ctrl에서 FSM수행이 끝나면 iCpDone신호를 받아 Interrupt 실행
- CpByteSize를 통해 Byte갯수 정보를 보내 그만큼만 유효 데이터로 실행
  나머지는 Zero Pedding을 통해 0으로 채움

- Cp_WrDtConv

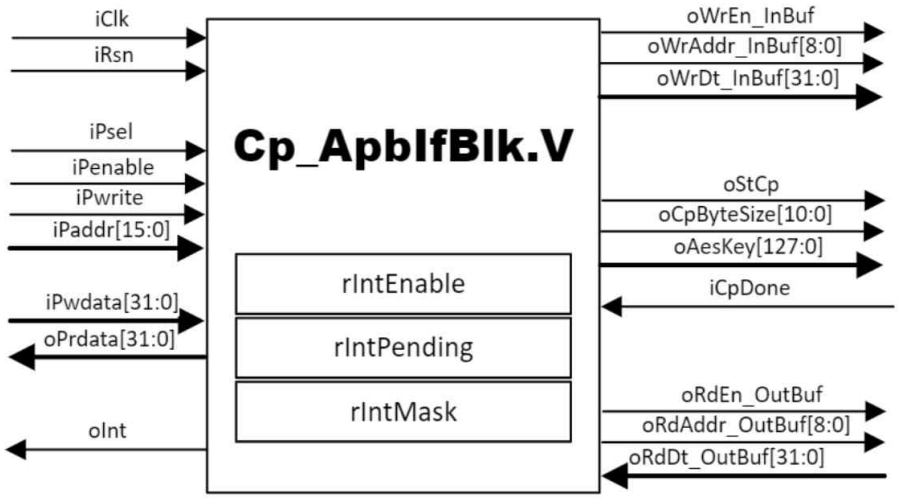
- (iPsel == 1'b1) && (iPenable == 1'b0) && (iPwrite == 1'b1) && (iPaddr[15:11] == 4'b0100) 일 때 Write신호를 발생
- 16bit Address를 9bit Address로 변환, iPaddr[10:2]으로 9bit Address 변환
  Ex) 16'h4000 --> 9'h000, 16'h4004 --> 9'h001 ... 16'h47FC --> 9'h1FF (511)
- Address에 맞는 Data를 전송

- Cp_RdDtConv

- (iPsel == 1'b1) && (iPenable == 1'b0) && (iPwrite == 1'b0) && (iPaddr[15:11] == 4'b0110) 일 때 Read신호를 발생
- 16bit Address를 9bit Address로 변환, iPaddr[10:2]으로 9bit Address 변환
  Ex) 16'h6000 --> 9'h000, 16'h6004 --> 9'h001 ... 16'h67FC --> 9'h1FF (511)
- Address에 맞는 Data를 전송받음

- AesCore

- Address가 0x2000 ~ 0x200C 일 때 AesCore로 iAesKey Data를 전송



1.3.2. Cp_Ctrl

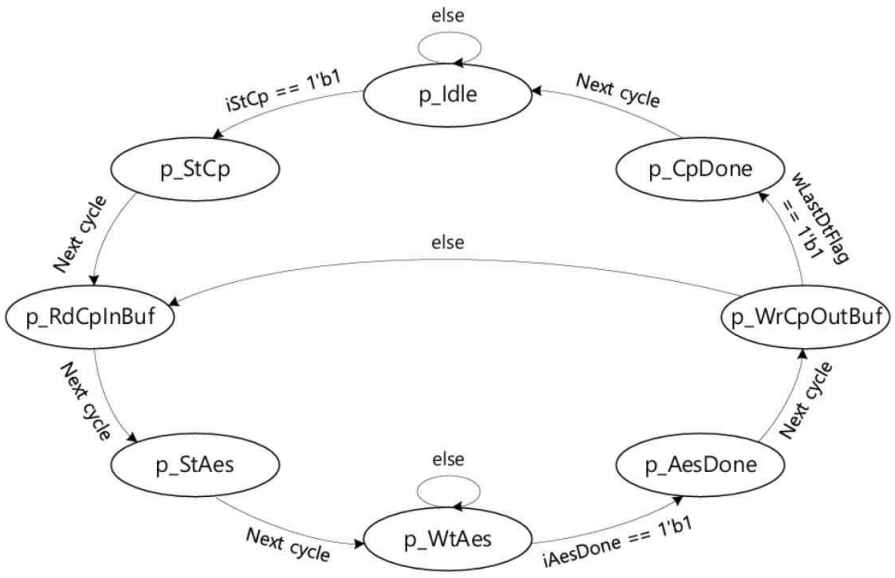
Description

InBuf에서 받은 Data를 Little Endian -> Big Endian으로 변환
Data를 AesCore모듈에 전송하고, iStAes신호를 전송하여 암호화
암호화된 Data를 전송받아 Big Endian --> Little Endian으로 변환 후 OutBuf로 전송

위와 같은 동작을 반복하는 FSM을 설계하여 동작
동작이 끝난 후 CpDone신호를 Apb로 전송

FSM

- p_Idle : 초기 단계
- p_StCp : Apb에서 iStCp신호가 들어오면 동작 시작
- p_RdCpInBuf : InBuf에서 Data를 Copy하여 내부에 저장
- p_StAes : AesCore의 동작 시작
- p_WtAes : AesCore Waiting --> iAesDone신호를 받을 때 까지 반복
- p_AesDone : AesCore의 동작이 끝나 iAesDone신호를 받아 넘어감
- p_WrCpOutBuf : OutBuf로 Data를 보냄, wLastDtFlag신호가 뜰 때까지 Aes반복
- p_CpDone : FSM 동작 완료 및 CpDone신호 출력



Connect Interface

- ApbIfBlk

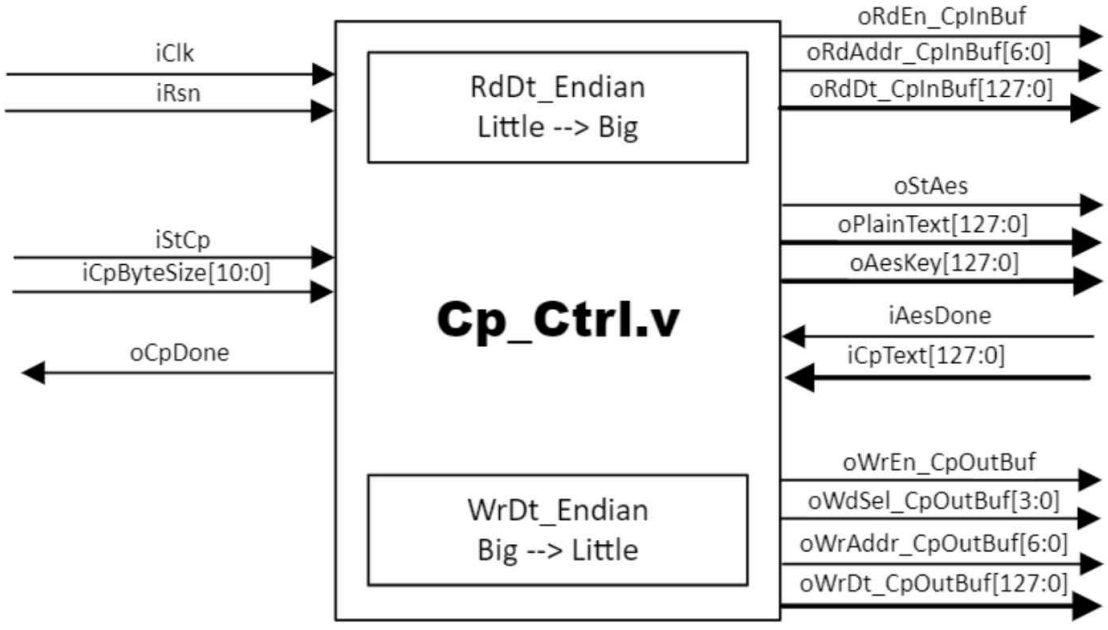
- StCp신호를 받아 FSM동작을 실행
- CpByteSize로 받은 횟수 / 16 만큼 동작을 반복
- FSM동작이 끝나면 CpDone신호를 전송

- AesCore

- StAes신호를 전송하여 AesCore동작을 실행
- Data를 보내 암호화 실행
- 동작이 끝나면 AesDone신호를 받아 전송 받은 Data를 OutBuf로 이동

- Cp_In/OutBuf

- Address에 맞는 Data를 Write/Read
- FSM동작에 따라 Data를 전송받고, 전송
- Endian Conversion을 통해 Data를 가공



1.3.3. AesCore

Description

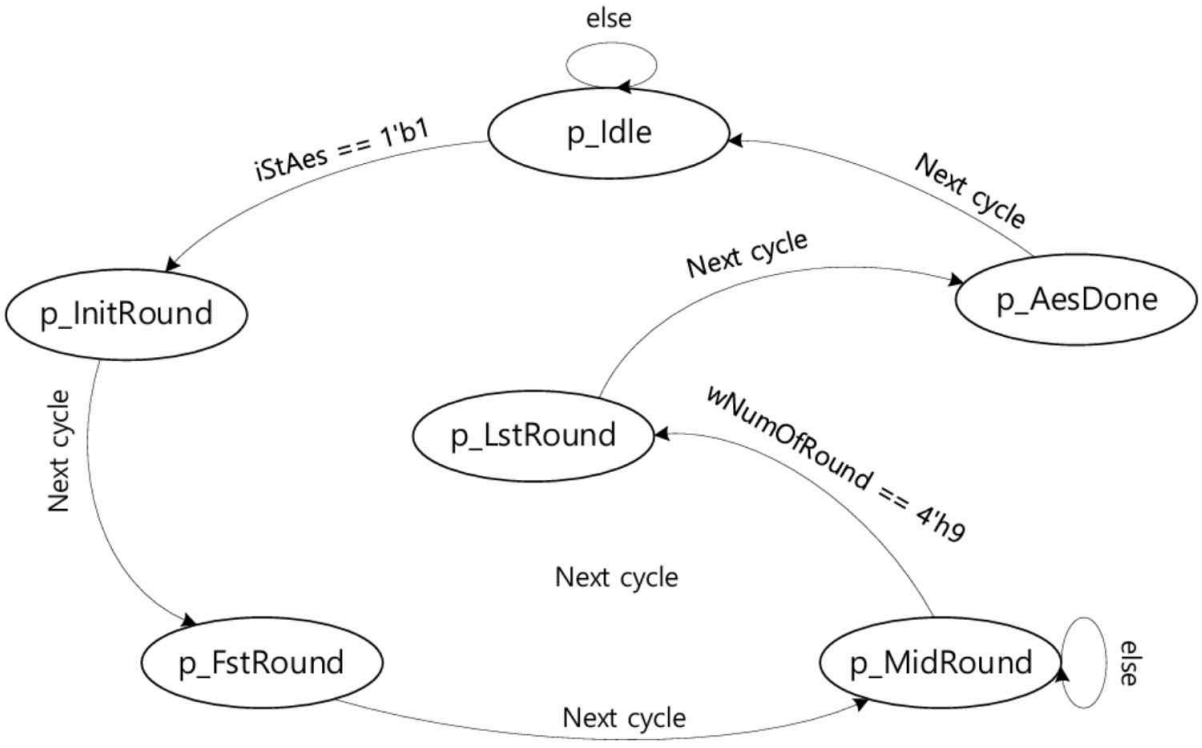
Ctrl에서 StAes신호를 받으면 전송받은 Data와 Key를 받아 Round Function을 진행

Round Funtion은 SubBytes / ShiftRows / MixColumns / AddRoundKey로 구성됨

AES-128같은 경우에는 1~9 Round는 위와 같이 진행하나 마지막 10Round는 MixColumns가 빠진 Round Function으로 진행됨

FSM

- p_Idle : 초기 단계
- p_InitRound : AddRoundKey단계만 실행 / Data와 Key를 불러오는 과정
- p_FstRound : 첫번째 Round 실행
- p_MidRound : 2~9번째 Round실행
- p_LstRound : 마지막 Round실행 / MixColumns 단계 제외
- p_AesDone : FSM 동작 완료



Connect Interface

- ApbIfBlk

- Initial Key를 전송받음

- Ctrl

- StAes신호를 전송받아 AesCore동작을 실행
- Key와 Data를 받아 암호화 과정 실행 --> FSM으로 작성
- 동작이 끝나면 AesDone신호와 암호화된 데이터를 전송

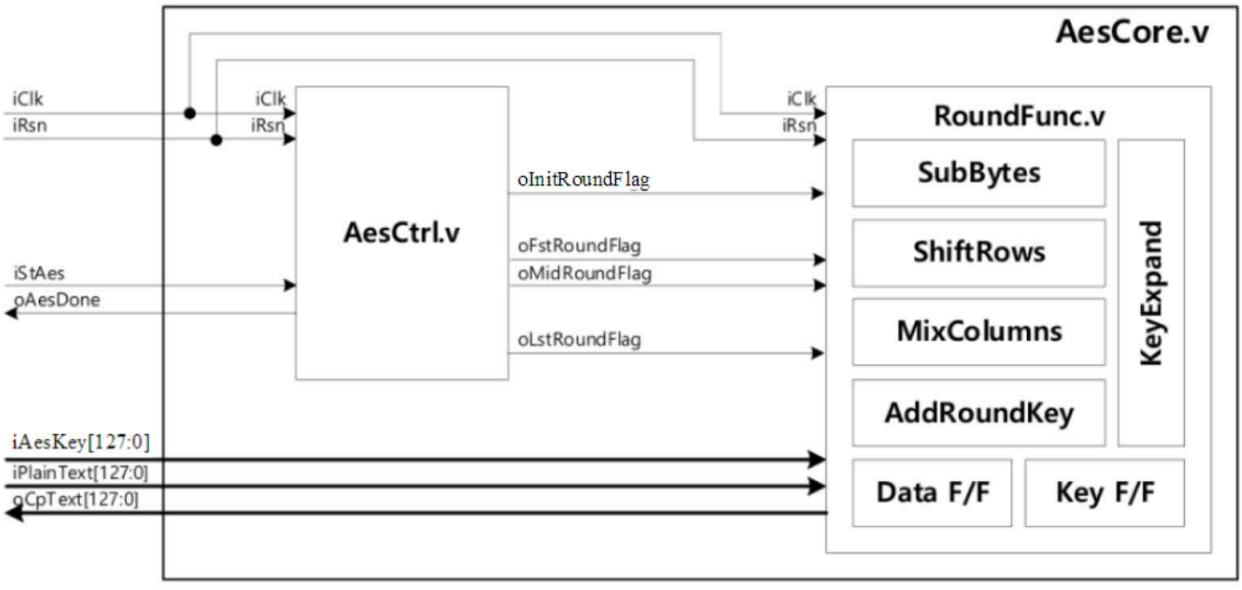
Internal Module

- AesCtrl

- StAes신호를 받아 FSM실행
- Round Signal를 통해 RoundFunc모듈을 조작

- RoundFunc

- SubBytes : S-Box에 따라 데이터를 치환 (비선형변환)
- ShiftRows : 데이터를 행 방향으로 Shift하는 과정 / 열 마다 Shift하는 횟수가 다름
- MixColumns : 각 열에 대한 행렬 연산으로 데이터를 변환 (선형변환)
- AddRoundKey : Key Data를 받아 Xor연산을 통해 Data 변환
- KeyExpand
  - RotWrod : 마지막 Word에서 한 칸 위쪽 순환 이동
  - SubWord : S-Box에 따라 Key를 치환 (SubByte와 같음)
  - RCon : 각 Round별 적용되는 상수 Table의 해당 값으로 Xor 연산을 하여 변환



1.3.4. Cp_Wr/RdDtConv

Description / Connect Interface

- Cp_WrDtConv

Apb에서 받은 32bit Data를 128bit로 변환하여 Address에 맞게 InBuf에 저장

- ApbIfBlk

- 16bit Address를 9bit Address로 변환, iPaddr[10:2]으로 9bit Address 변환
  Ex) 16'h4000 --> 9'h000, 16'h4004 --> 9'h001 ... 16'h47FC --> 9'h1FF (511)
- Address에 맞는 Data를 전송받음

- InBuf

- Address에 맞는 Data를 전송하여 저장

- Cp_RdDtConv

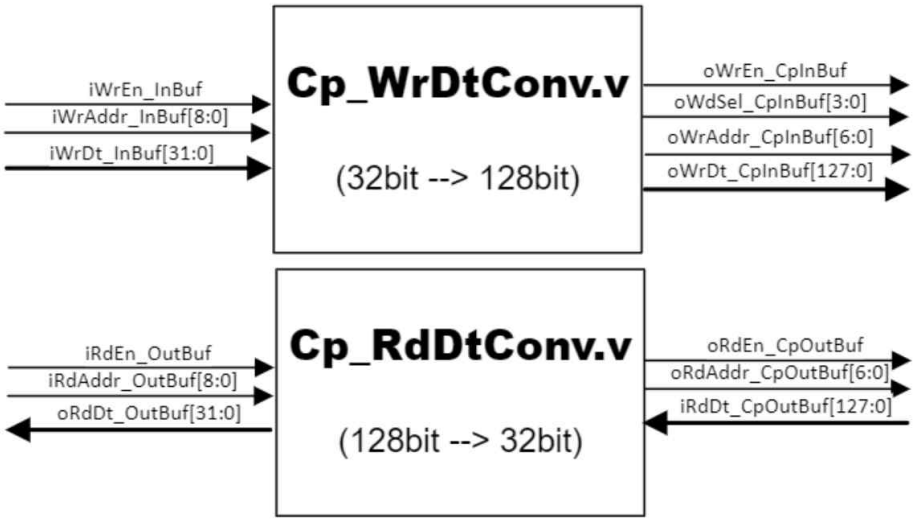
OutBuf에서 받은 128bit데이터를 32bit로 변환하여 Address에 맞게 Apb로 전송

-ApbIfBlk

- 16bit Address를 9bit Address로 변환, iPaddr[10:2]으로 9bit Address 변환
  Ex) 16'h6000 --> 9'h000, 16'h6004 --> 9'h001 ... 16'h67FC --> 9'h1FF (511)
- Address에 맞는 Data를 전송

- OutBuf

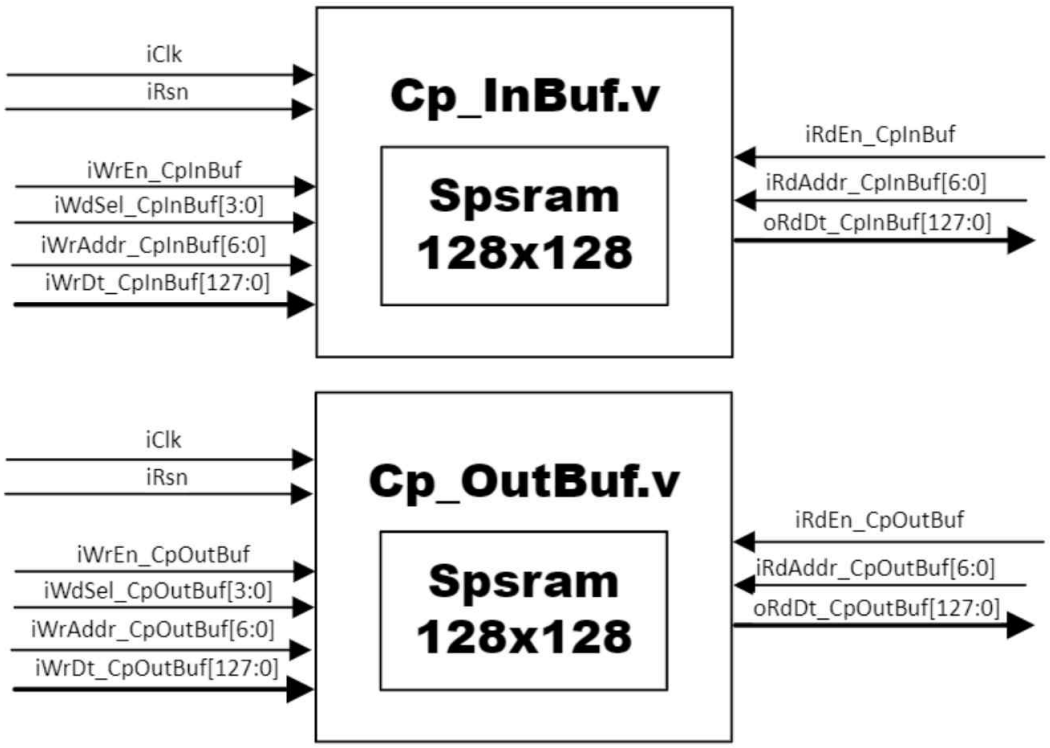
- Address에 맞는 저장된 Data를 전송받음



1.3.5. In/OutBuf

Description

내부 Spsrma에 Data를 저장하는 버퍼
In/OutBuf 각각 Wr/RdDtConv와 연결되어 Address에 맞는 Data를 주고받음



1.4. Further work

AesCore, Controller및 Buffer가 하나의 Clock신호를 받아 동작하고 있다. 이는 각 모듈이 사용되지 않을 때에도 전력을 소비하기 때문에 각 모듈별로 Clock Gating을 통해 각각의 Enable신호에 맞게 Clock이 작동되게 만들다면 전력 소비를 줄일 수 있을 것이다.

암호화를 시키면서 생성된 총 11라운드의 Key를 저장해 놓는다면 후에 복호화를 시킬 때 KeyExpansion과정이 생략될 수 있을 것이다.

현재 구현화시킨 AES-128모델은 암호화 과정만 진행하는 모델이다. 시간이 좀 더 주어진다면 복호화 과정까지 구현하여 AesCore모듈에 Mux신호를 이용해서 신호에 따라 암호화 및 복호화 과정을 진행하는 모델을 구현해해 볼 수 있을 것 같다.

1.5. Appendix

// AesCore 모듈을 설계할 때 참고한 Round 알고리즘 코드
https://github.com/michaelehab/AES-Verilog

// 전체적인 AES동작 방식을 확인하기 위한 영상
https://youtube.com/playlist?list=PLzQqHxFtQGyfkgcI2Ya9oDQ4K8J4w0T5x&si=DM5rjexYbwxc8sxP
https://www.youtube.com/watch?v=mlzxpkdXP58

1.6. Timing Diagram