**ASE-GCM core Description**

1. Mode:

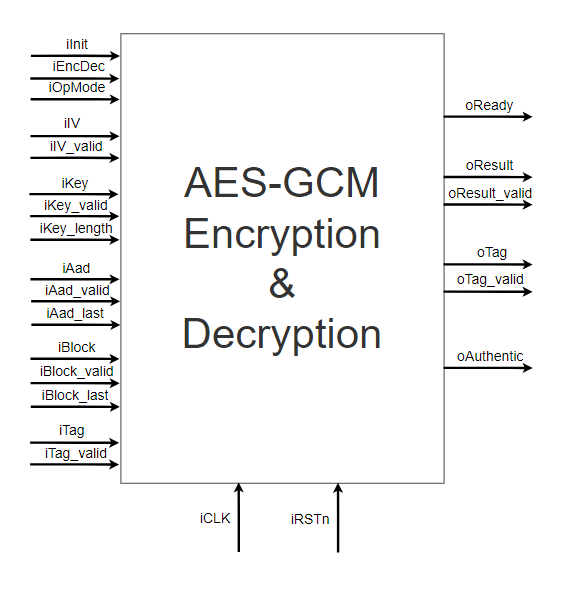
* AES-GCM or AES only
* Both Encryption and Decryption
* Key length: 128-bit or 256-bit

1. Input:

* Key,
* IV (96-bit)
* Block data (Plaintext/Ciphertext)
* Additional Authentication Data (AAD)
* Tag (in Decryption mode)

1. Output:

* Result (Ciphertext/Plaintext)
* Tag (in Encryption mode)
* Authentication (in Decryption mode)



|  |  |  |  |
| --- | --- | --- | --- |
| PIN | Direction | Width | Description |
| iClk | Input | 1 | Clock |
| iRstn | Input | 1 | Negative edge reset: reset after using the core |
| iInit | Input | 1 | Initialize signal: assert when using the core |
| iEncDec | Input | 1 | Assert for Encryption, deassert for Decryption |
| iOpMode | Input | 1 | Operation Mode: assert for AES-GCM, deassert for AES only |
| oReady | Output | 1 | Output ready signal: input new data when ready is asserted |
| iIV | Input | 96 | 96-bit length IV |
| iIV\_valid | Input | 1 | When asserted, IV is valid |
| iKey | Input | 256 | Key |
| iKey\_valid | Input | 1 | When asserted, Key is valid |
| iKey\_len | Input | 1 | When asserted, Key length is 256-bit.  When deasserted, Key length is 128-bit. |
| iAad | Input | 128 | Additional Authentic Data or Length(A,C): when Ready = 1, change data every clock cycle |
| iAad\_valid | Input | 1 | When asserted, AAD is valid. |
| iAad\_last | Input | 1 | Asserted when input the last AAD data. |
| iBlock | Input | 128 | Input Plaintext or Ciphertext. |
| iBlock\_valid | Input | 1 | When asserted, Plaintext or Ciphertext is valid. |
| iBlock\_last | Input | 1 | Asserted when input the last Block data. |
| iTag | Input | 128 | Input Tag for authentication in Decryption mode |
| iTag\_valid | Input | 1 | When asserted, Tag is valid |
| oResult | Output | 128 | Output Ciphertext or Ciphertext |
| oResult\_valid | Output | 1 | When asserted, Result is valid |
| oTag | Output | 128 | Output Tag in Encryption mode. |
| oTag\_valid | Output | 1 | When asserted, output Tag is valid |
| oAuthentic | Output | 1 | When asserted, indicating authentic Block in Decryption mode. |

**AES-GCM Testbench**

**Testcase 1:** AES-GCM Encryption 256-bit Key with both AAD and Plaintext

KEY: 256'hE3C08A8F06C6E3AD95A70557B23F75483CE33021A9C72B7025666204C69C0B72

BLOCK: 128'h08000F101112131415161718191A1B1C

128'h1D1E1F202122232425262728292A2B2C

128'h2D2E2F303132333435363738393A0002

AAD: 128'hD609B1F056637A0D46DF998D88E52E00

128'hB2C2846512153524C0895E81\_00000000

IV: 96'h12153524C0895E81B2C28465

LEN(A,C): 128'h00000000000000E0\_0000000000000180

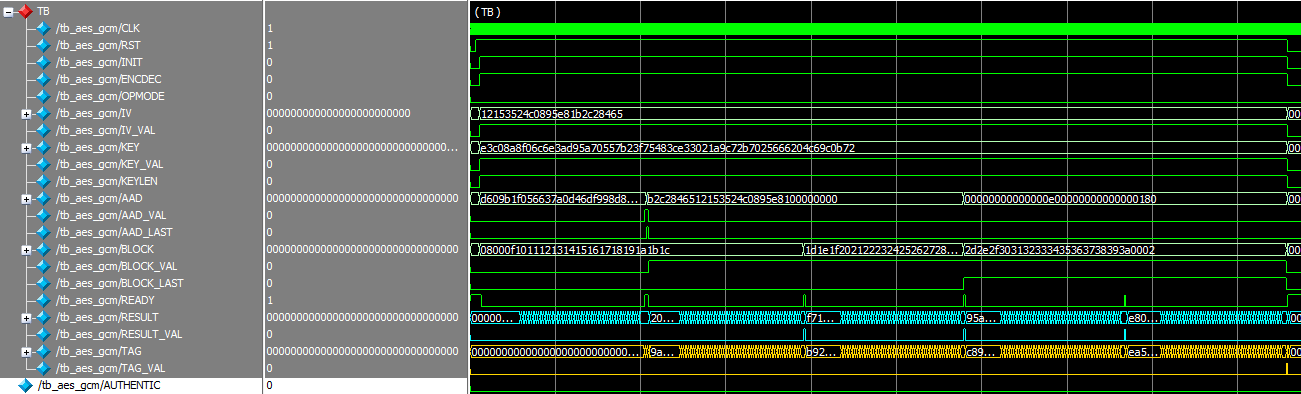
**Expected Outputs:**

CIPHER: 128'hE2006EB42F5277022D9B19925BC419D7

128'hA592666C925FE2EF718EB4E308EFEAA7

128'hC5273B394118860A5BE2A97F56AB7836

TAG 128'h5CA597CDBB3EDB8D1A1151EA0AF7B436



**Testcase 2:** AES-GCM Encryption 128-bit Key with both AAD and Plaintext

KEY: {128'hAD7A2BD03EAC835A6F620FDCB506B345, 128'd0}

BLOCK: 128'h08000F101112131415161718191A1B1C

128'h1D1E1F202122232425262728292A2B2C

128'h2D2E2F303132333435363738393A0002

AAD: 128'hD609B1F056637A0D46DF998D88E52E00

128'hB2C2846512153524C0895E81\_00000000

IV: 96'h12153524C0895E81B2C28465

LEN(A,C): 128'h00000000000000E0\_0000000000000180

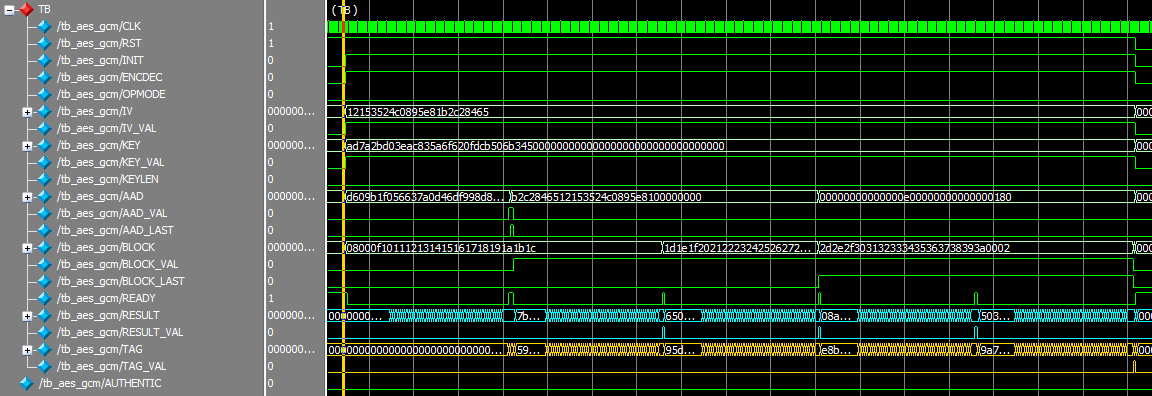
**Expected Outputs:**

CIPHER: 128'h701AFA1CC039C0D765128A665DAB6924

128'h3899BF7318CCDC81C9931DA17FBE8EDD

128'h7D17CB8B4C26FC81E3284F2B7FBA713D

TAG 128'h4F8D55E7D3F06FD5A13C0C29B9D5B880



**Testcase 3:** AES-GCM Encryption 256-bit Key with only Plaintext

KEY: 256'hFEFFE9928665731C6D6A8F9467308308\_FEFFE9928665731C6D6A8F9467308308

BLOCK: 128'hD9313225F88406E5A55909C5AFF5269A

128'h86A7A9531534F7DA2E4C303D8A318A72

128'h1C3C0C95956809532FCF0E2449A6B525

128'hB16AEDF5AA0DE657BA637B391AAFD255

IV: 96'hCAFEBABEFACEDBADDECAF888

LEN(A,C): 128'h00000000000000000000000000000200

**Expected Outputs:**

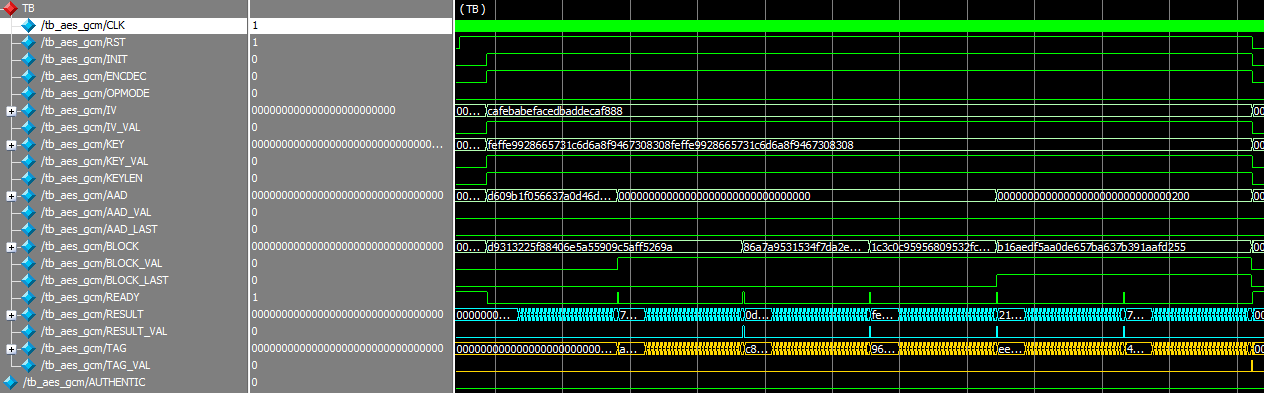
CIPHER: 128'h522DC1F099567D07F47F37A32A84427D

128'h643A8CDCBFE5C0C97598A2BD2555D1AA

128'h8CB08E48590DBB3DA7B08B1056828838

128'hC5F61E6393BA7A0ABCC9F662898015AD

TAG 128'hB094DAC5D93471BDEC1A502270E3CC6C



**Testcase 4:** AES-GCM Encryption 256-bit Key with only AAD

KEY: 256'h83C093B58DE7FFE1C0DA926AC43FB360\_9AC1C80FEE1B624497EF942E2F79A823

AAD: 128'h84C5D513D2AAF6E5BBD2727788E52300

128'h8932D6127CFDE9F9E33724C608000F10

128'h1112131415161718191A1B1C1D1E1F20

128'h2122232425262728292A2B2C2D2E2F30

128'h3132333435363738393A3B3C3D3E3F00

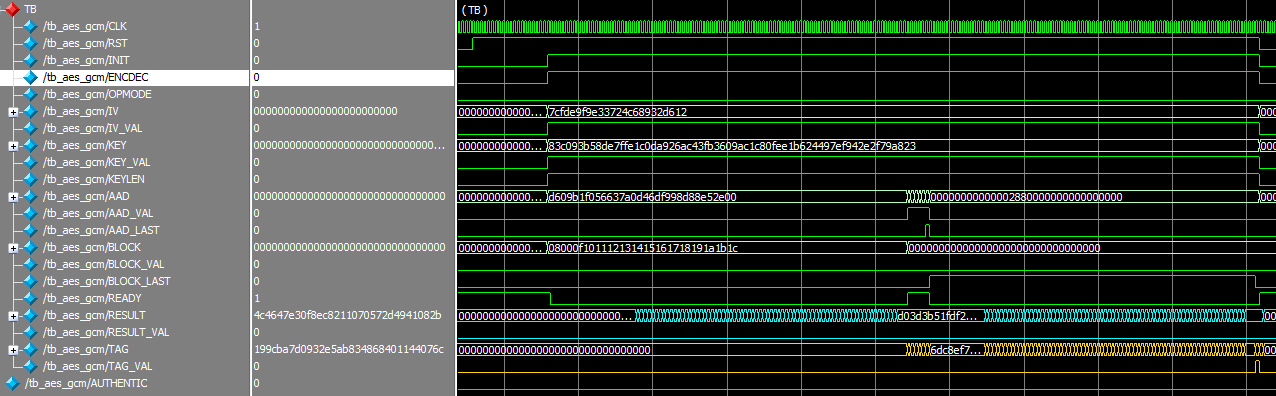
128'h05000000000000000000000000000000

IV: 96'h7CFDE9F9E33724C68932D612

LEN(A,C): 128'h0000000000000288\_0000000000000000

**Expected Outputs:**

TAG 128'h6EE160E8FAECA4B36C86B234920CA975



**Testcase 5:** AES-GCM Decryption 256-bit Key with both AAD and Plaintext

KEY: 256'hE3C08A8F06C6E3AD95A70557B23F75483CE33021A9C72B7025666204C69C0B72

BLOCK: 128'hE2006EB42F5277022D9B19925BC419D7

128'hA592666C925FE2EF718EB4E308EFEAA7

128'hC5273B394118860A5BE2A97F56AB7836

AAD: 128'hD609B1F056637A0D46DF998D88E52E00

128'hB2C2846512153524C0895E81\_00000000

IV: 96'h12153524C0895E81B2C28465

LEN(A,C): 128'h00000000000000E0\_0000000000000180

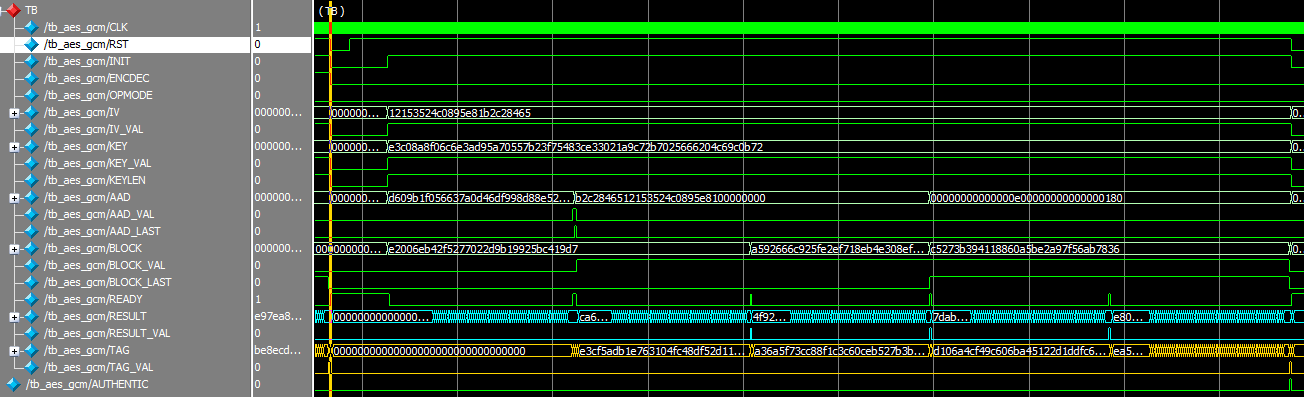
**Expected Outputs:**

CIPHER: 128'h08000F101112131415161718191A1B1C

128'h1D1E1F202122232425262728292A2B2C

128'h2D2E2F303132333435363738393A0002

TAG 128'h5CA597CDBB3EDB8D1A1151EA0AF7B436



**Testbench Signal Descriptions**

1. Control signals:

* RST: reset LOW. Perform reset after finish using the core
* INIT: Initialize the core. Set to HIGH when input data to the core, and keep HIGH the whole time when using the core. Set to LOW after finish using the core.
* ENCDEC: HIGH – Perform encryption. LOW - Perform decryption.
* OPMODE: operation mode. HIGH – AES only. LOW – AES-GCM.
* KEYLEN: indicate the length of the input key. HIGH – 256b key. LOW – 128b key.
* AAD\_LAST: set to HIGH when input the last AAD data.

1. Input data signals:

* KEY: input key
* KEY\_VAL: indicate valid input key. Set to HIGH when the input key is valid.
* AAD: input AAD or input length (len(A,C)).
* AAD\_VAL: indicate valid input AAD. Set to HIGH when the input AAD is valid. When input len(A,C), don’t care about this signal
* BLOCK: input block.
* BLOCK\_VAL: indicate valid input block. Set to HIGH when the input block is valid.
* BLOCK\_LAST:

1. Output signals:

* READY: the core is ready to accept input data. Input data or change data to new one when this signal is HIGH.
* RESULT: output ciphertext or plaintext
* RESULT\_VALID: indicate the output ciphertext or plaintext is valid. Read the value of RESULT when this signal is HIGH
* TAG: output tag
* TAG\_VALID: indicate the output tag is valid. Read the value of TAG when this signal is HIGH.

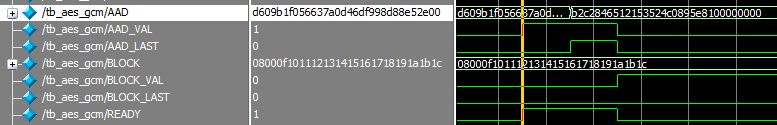
**Use Guide AES-GCM**

1. **With both AAD and Plaintext/Ciphertext**
2. After the core is reset, the READY signal is HIGH.
3. Set control signals (INIT, OPMODE, ENCDEC, KEYLEN), input KEY + valid key, and input IV + valid IV.

At this step, the core is calculating HashKey.

1. Wait for READY signal. At this step, the HashKey is calculated.
2. Input AAD + valid AAD (if input the last AAD, set the AAD\_LAST).

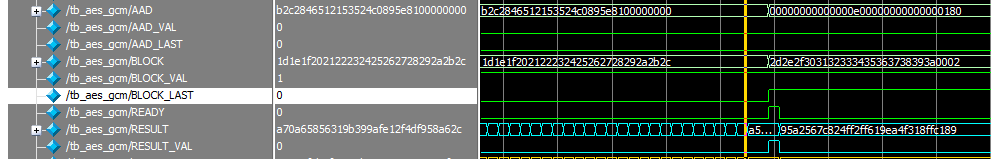
Note: Change each AAD data every clock rising edges.



1. After input the last AAD, input the first BLOCK data + valid BLOCK.

At this step, the core is calculating the first Ciphertext/Plaintext.

1. Wait for READY signal (or RESULT\_VAL). At this step, the first Ciphertext/Plaintext is calculated.
2. Repeat step 5 & 6 until the last BLOCK data (if input the last BLOCK, set the BLOCK\_LAST and input the length(A,C))



1. After input the last BLOCK, wait for READY signal (or RESULT\_VALID) to get the last Ciphertext/Plaintext.
2. Wait for TAG\_VAL signal to get the output TAG.
3. Reset the core
4. **With only Plaintext/Ciphertext**

Skip step 4 in **I**.

1. **With only ADD**

Do step 1-4 in **I**.

5. After input the last AAD, set BLOCK\_LAST signal (only this signal).

6. Wait for TAG\_VAL signal the get the output TAG.

7. Reset the core

**Use Guide AES only**

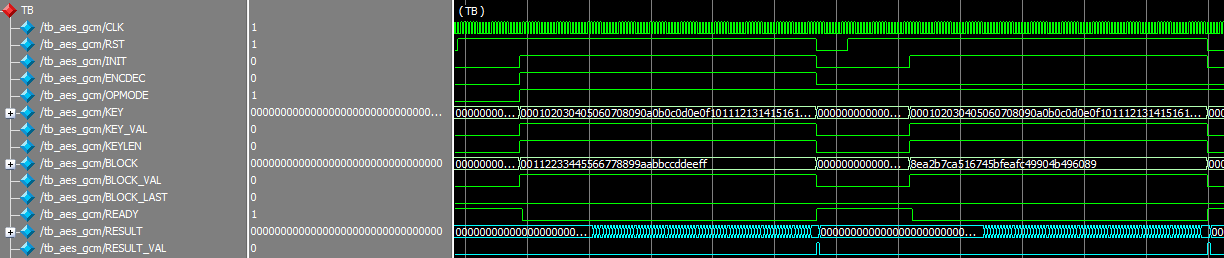
**Testcase 6 and 7: AES only in Encryption and Decryption mode**

KEY: 256'h000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f

BLOCK: 128'h00112233445566778899aabbccddeeff

CIPHER: 128'h8ea2b7ca516745bfeafc49904b496089

Note: Swap BLOCK and CIPHER for Decryption mode in testcase 7



Use guide:

1. After the core is reset, the READY signal is HIGH.
2. Set control signal (INIT, OPMODE, ENCDEC, KEYLEN), input KEY + valid KEY, input BLOCK + valid BLOCK
3. Wait for READY (or RESULT\_VAL) to get the Ciphertext/Plaintext.

**FPGA Results**

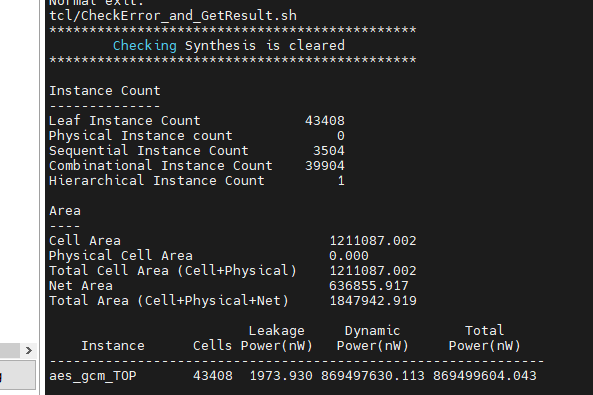
Device: Stratix IV EP4SGX530NF45C3

Fmax: 160.98MHz

|  |  |  |
| --- | --- | --- |
|  | Combinational ALUTs | Registers |
| aes-gcm | 11566 | 3504 |
| * gctr\_block | 3789 | 2987 |
| * + aes\_core | 3489 | 2469 |
| * + aes\_decipher\_block | 991 | 137 |
| * + aes\_encipher\_block | 609 | 137 |
| * + aes\_key\_mem | 1526 | 2191 |
| * + aes\_sbox | 160 |  |
| * ghash\_block | 7391 |  |
| * + gfmul | 7263 |  |

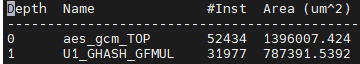
**ASIC Results**

Synthesis Results:

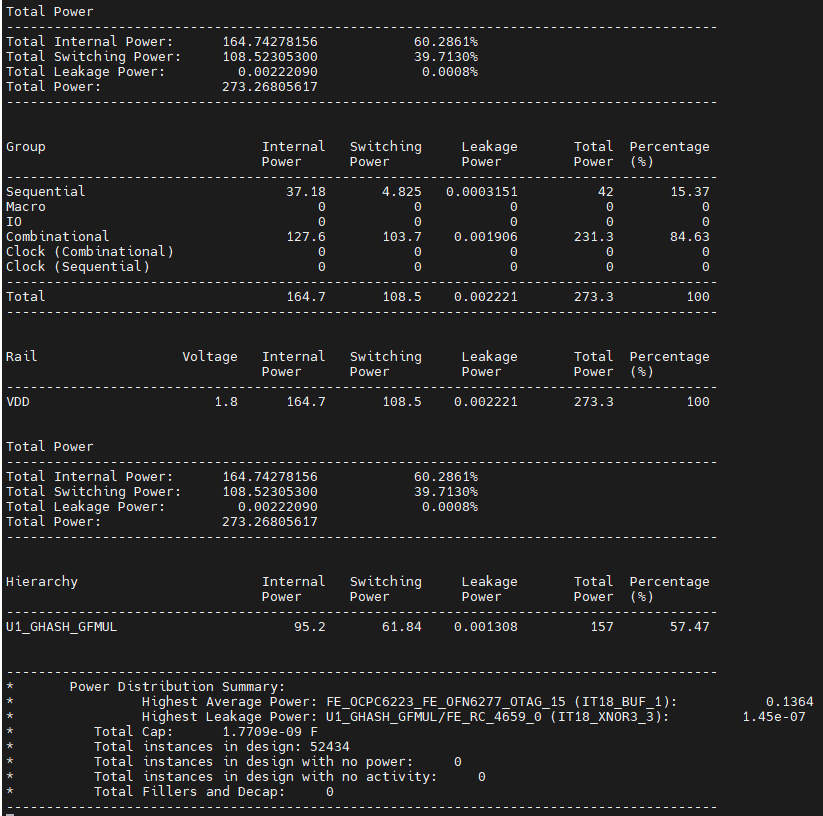




PnR Results:



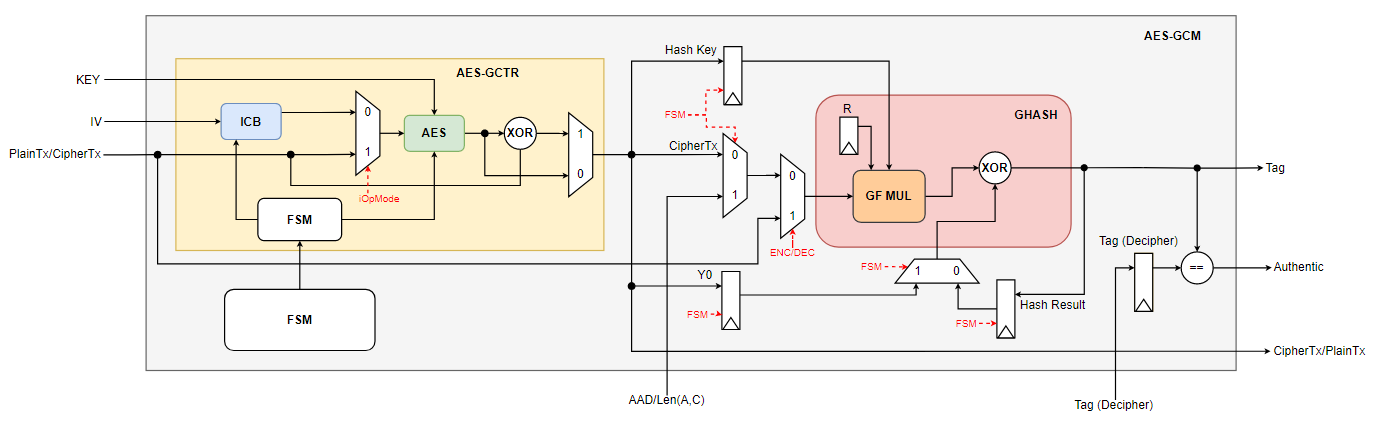






BLOCK DIAGRAM

and STATE MACHINE



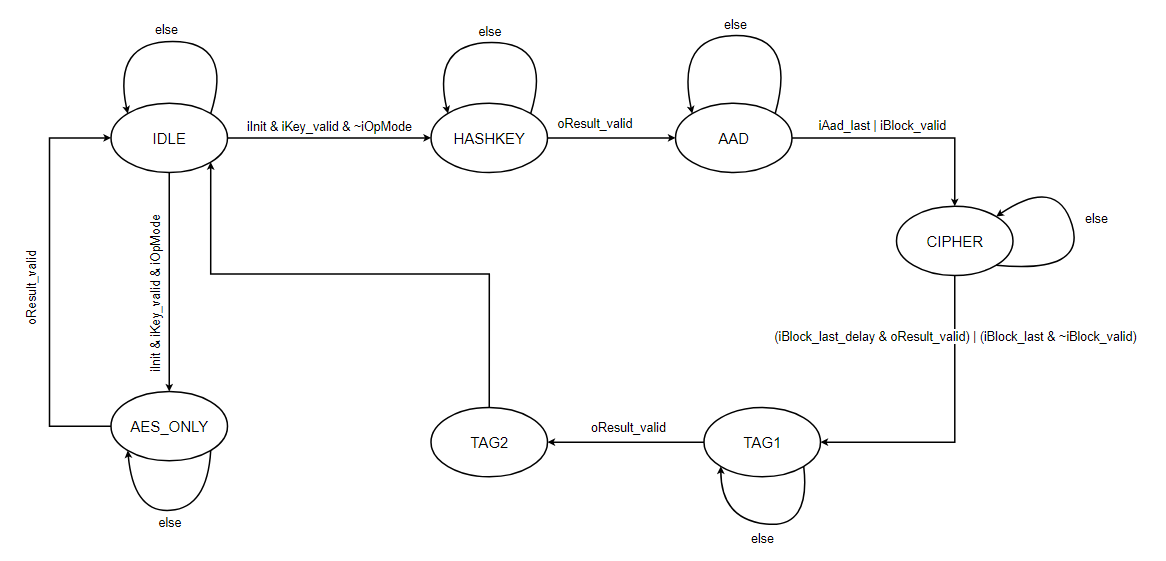
Figure 1. AES-GCM overview

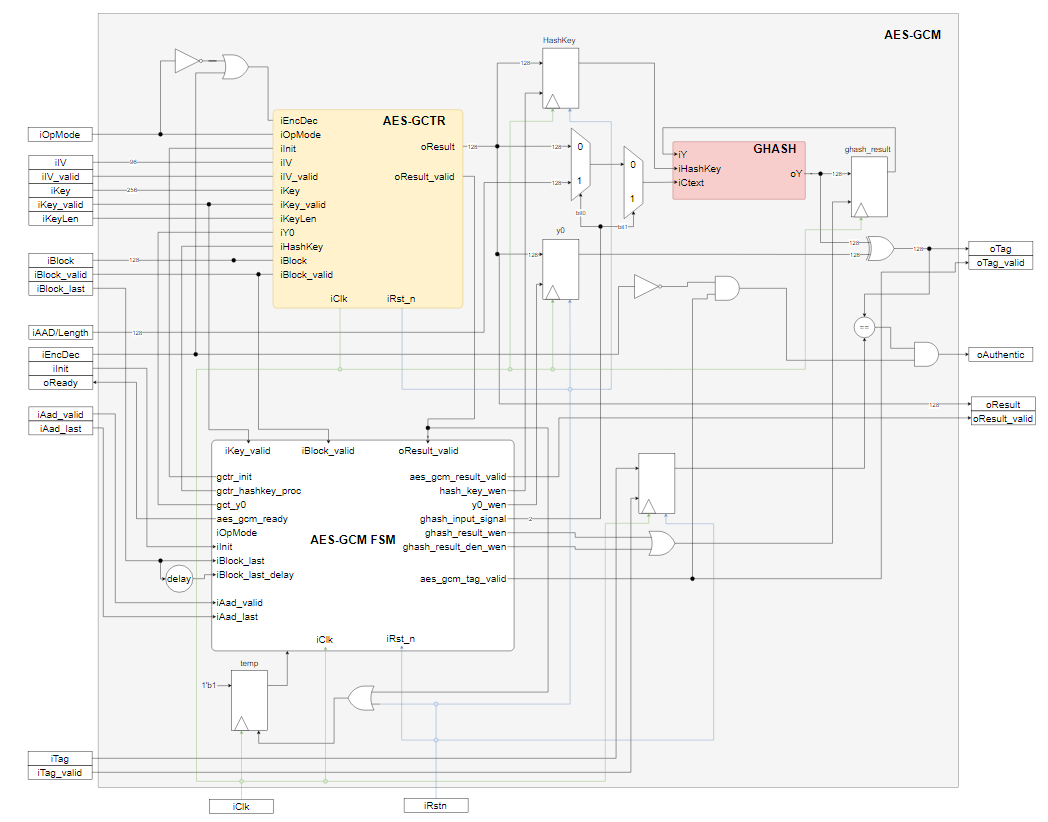
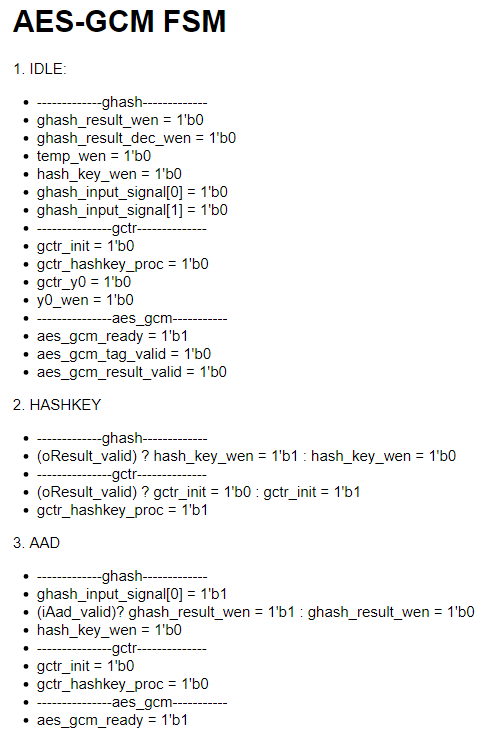
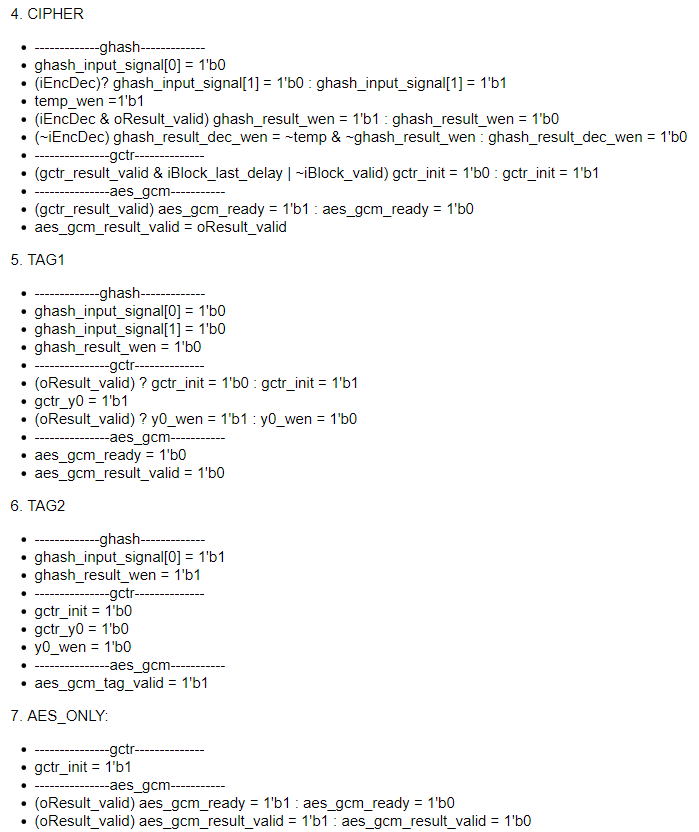
Figure 2. AES-GCM FSM

Figure 3. AES-GCM connection

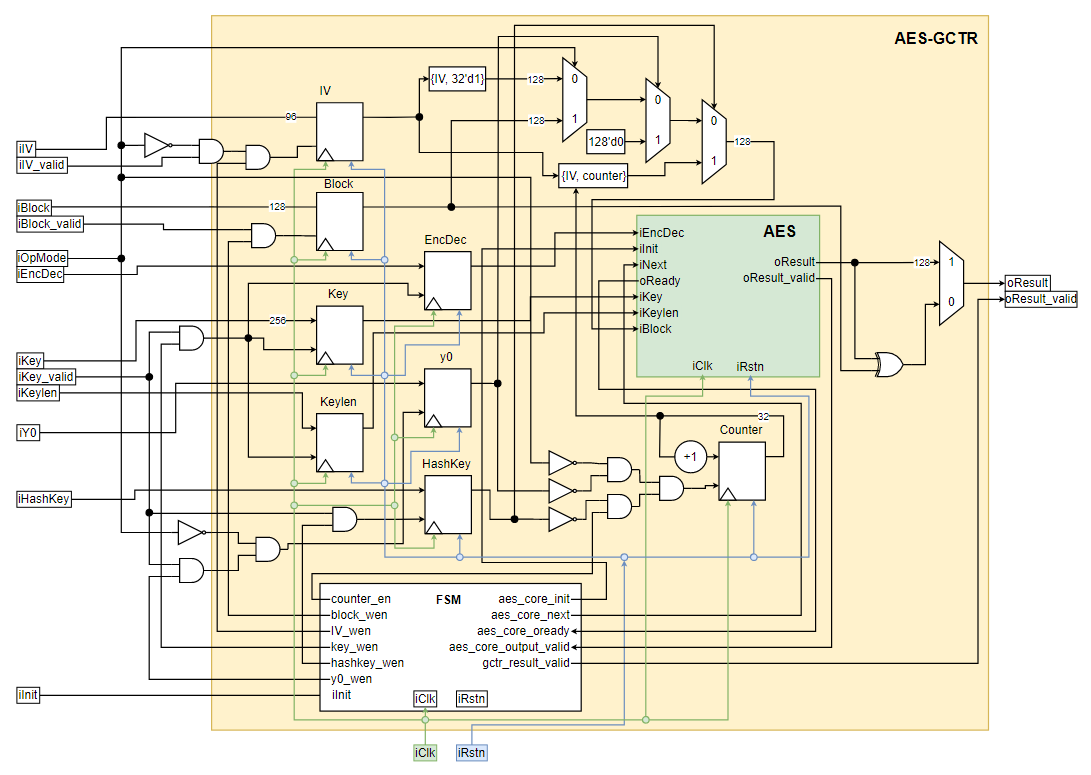


Figure 4. AES-GCTR connection

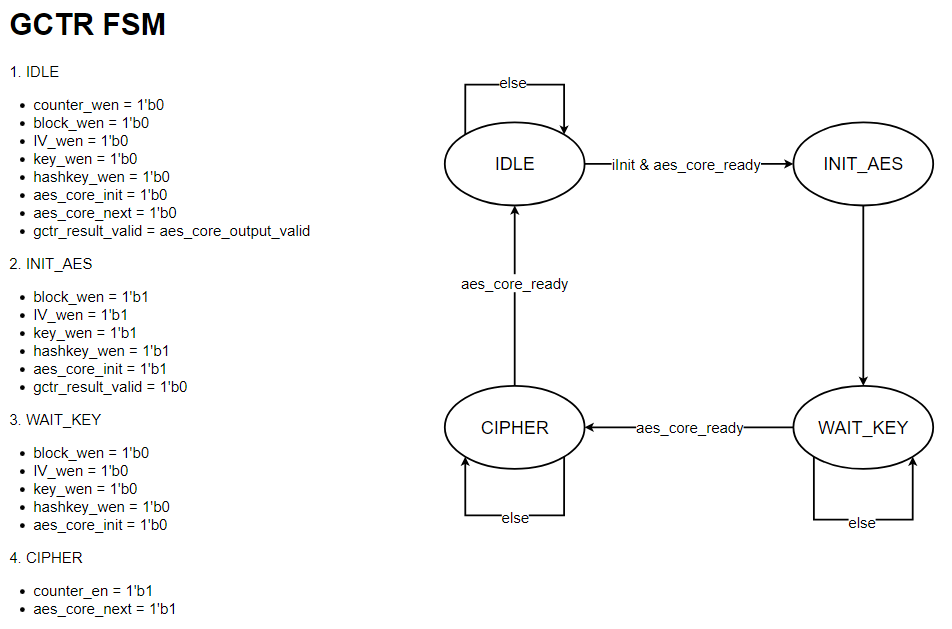


Figure 5. GCTR-FSM

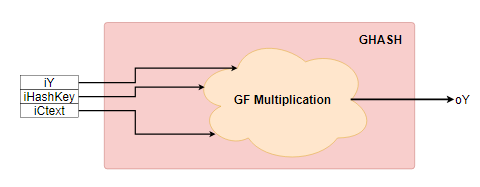


Figure 6. GHASH

Multiplication in ***GF(2128)***

Each element is a vector of 128 bits. The **ith** bit of an element **X** is denoted as **X­­i**. The leftmost bit is **X­­i** and the rightmost bit is X127. The multiplication operation uses the special element **R = 1110001||0**, and is defined in Algorithm 1. The argument **rightshift()** moves the bits of its argument one bit to the right. More formally, whenever **W** = **rightshift(V)**, then **Wi** = **Vi-1**for 1 <= i <= 127 and **W0** = 0.

What we want to compute: oY = gf\_mul(iHashKey, iCtext) xor iY

