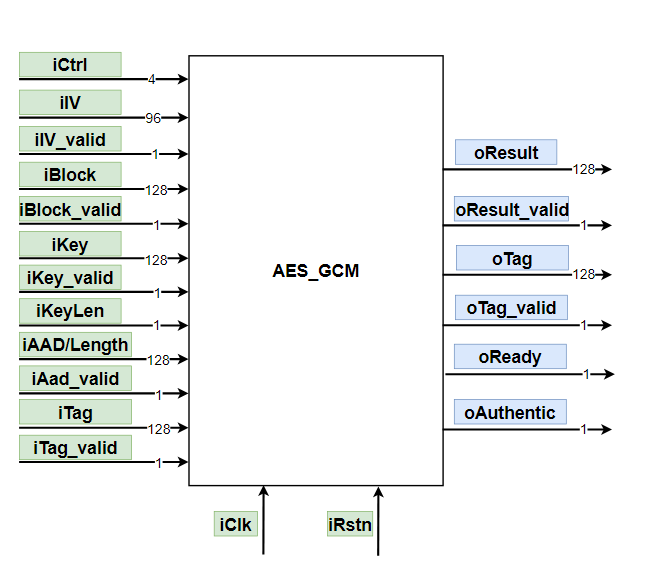
AES-GCM

BLOCK DIAGRAMs

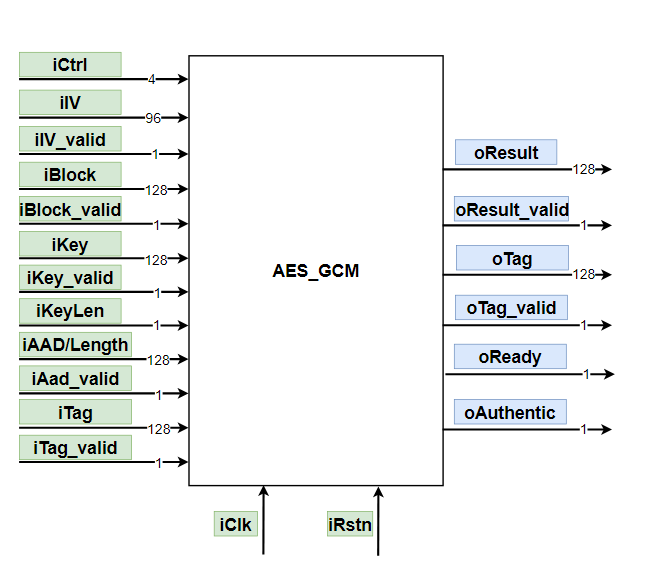
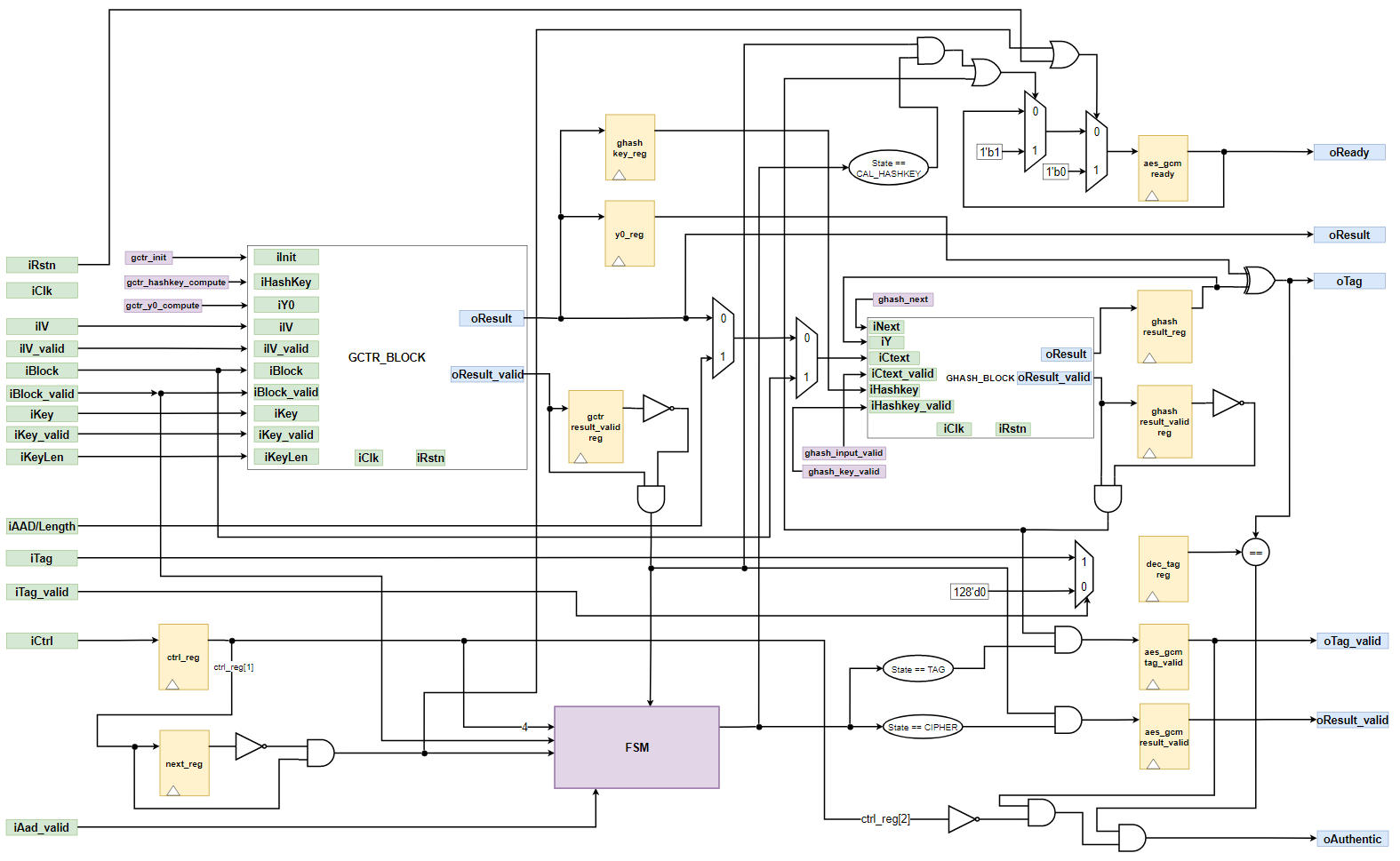
and STATE MACHINEs

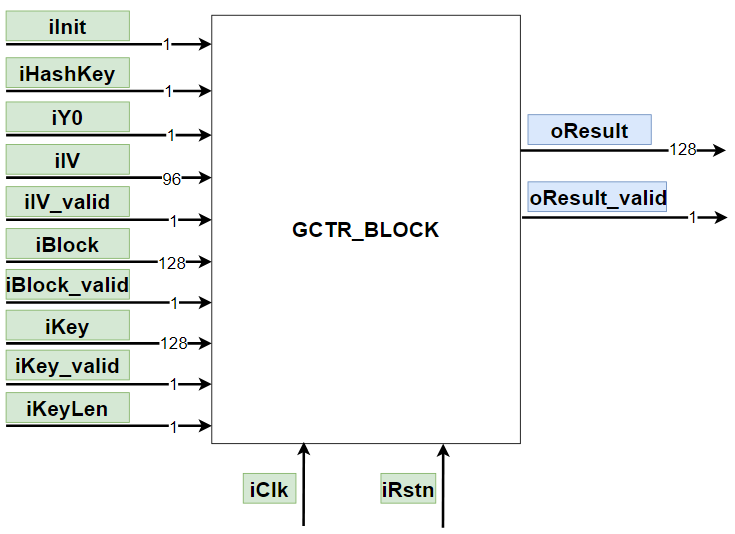


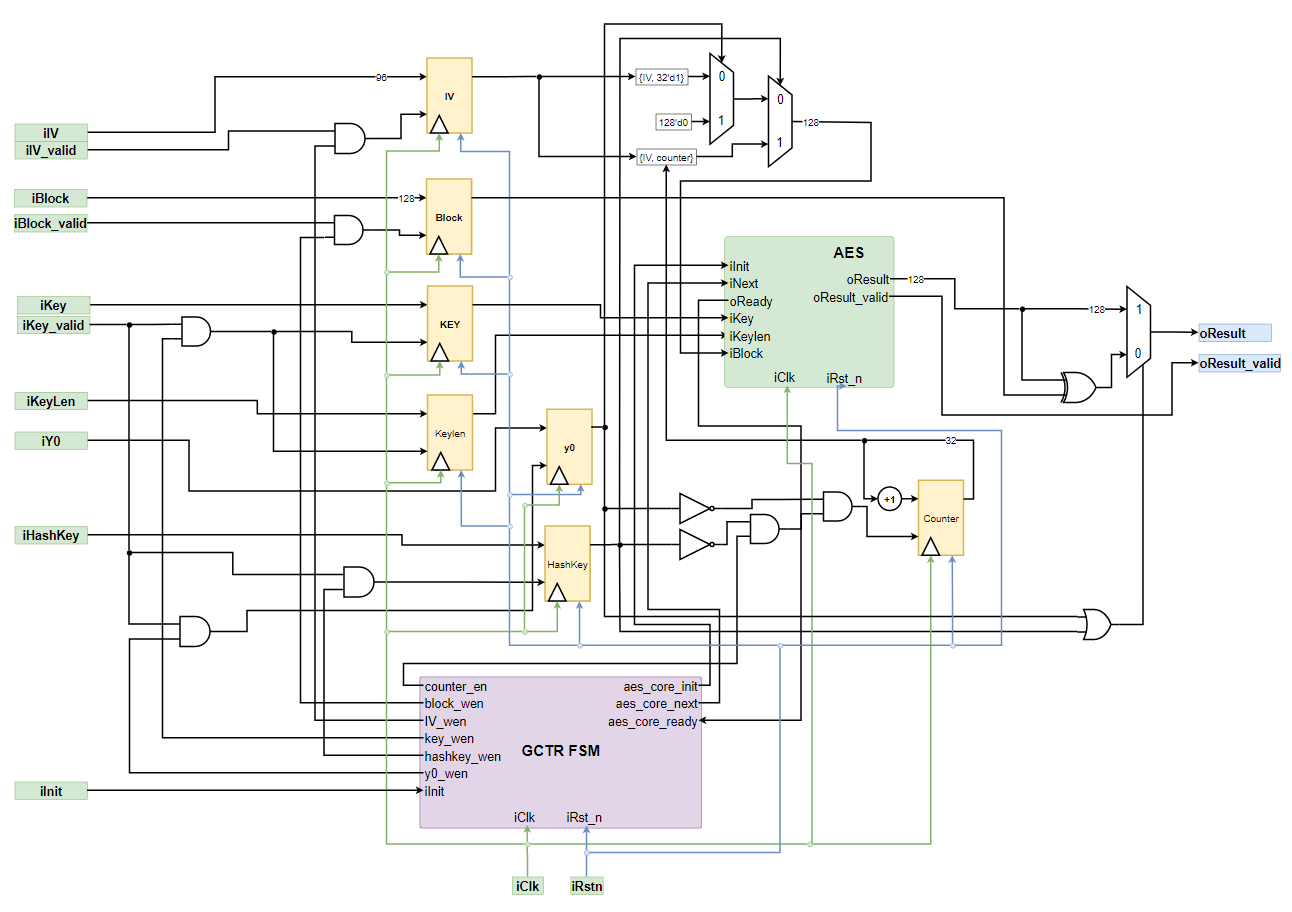
|  |  |  |  |
| --- | --- | --- | --- |
| PIN | Direction | Width | Description |
| iClk | Input | 1 | Clock |
| iRstn | Input | 1 | Negative edge reset: reset after using the core |
| CONTROL | | | |
| iCtrl | Input | ４ | Initialize signal: assert when using the core |
| oReady | Output | 1 | Output ready signal: input new data when ready is asserted |
| DATA | | | |
| 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. |
| iBlock | Input | 128 | Input Plaintext or Ciphertext. |
| iBlock\_valid | Input | 1 | When asserted, Plaintext or Ciphertext is valid. |
| iTag | Input | 128 | Input Tag for authentication in Decryption mode |
| iTag\_valid | Input | 1 | When asserted, Tag is valid |
| OUTPUT | | | |
| 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. |

|  |  |
| --- | --- |
| Name | AES\_GCM |
| File | aes\_gcm.v |

|  |  |
| --- | --- |
| Submodule | File |
| GCTR\_BLOCK | gctr\_block.v |
| GHASH\_BLOCK | ghash\_block.v |

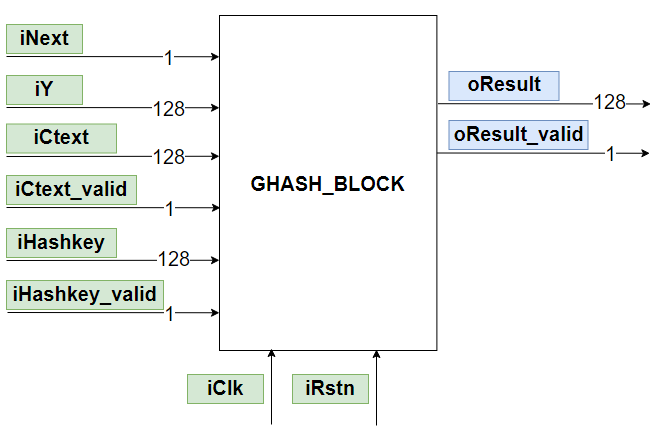


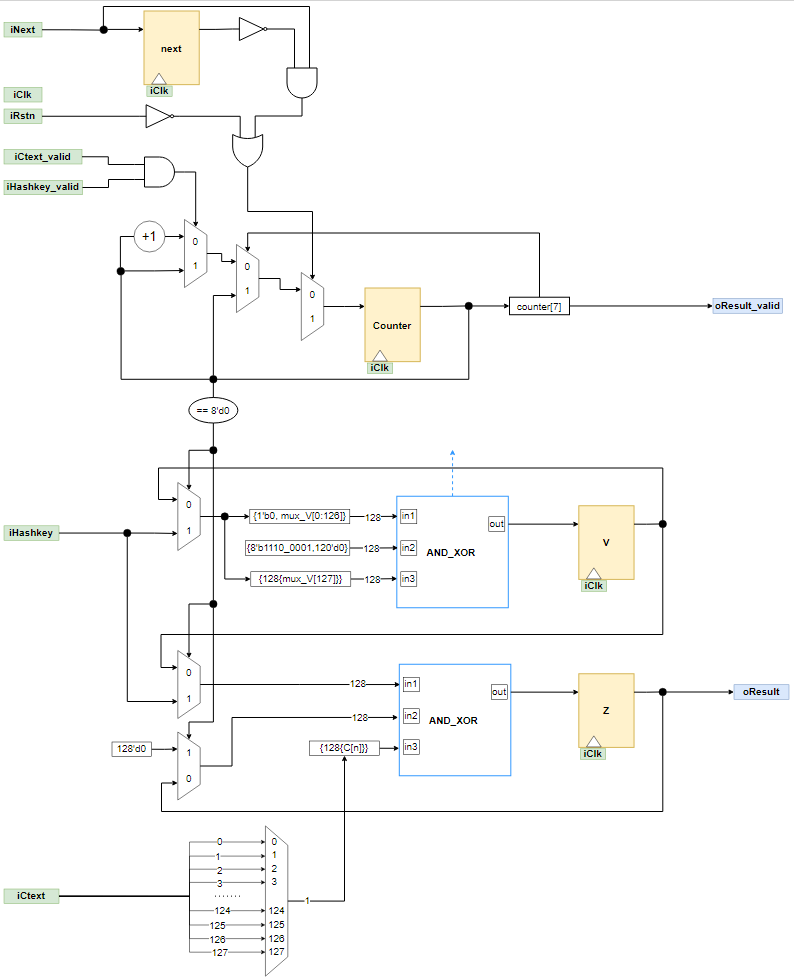


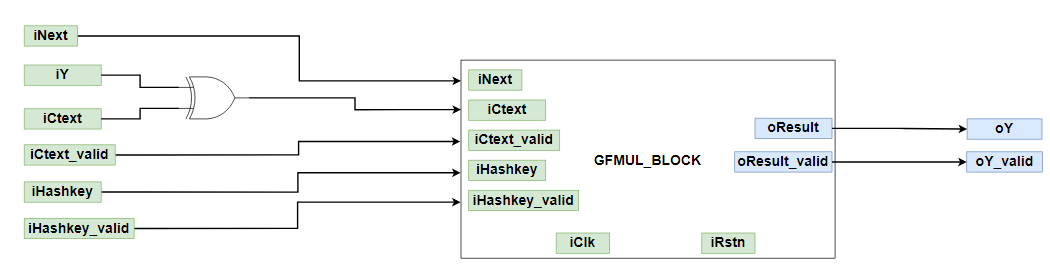


|  |  |
| --- | --- |
| Submodule | File |
| AES | aes\_core.v |

|  |  |
| --- | --- |
| Name | GCTR\_BLOCK |
| File | gctr\_block.v |



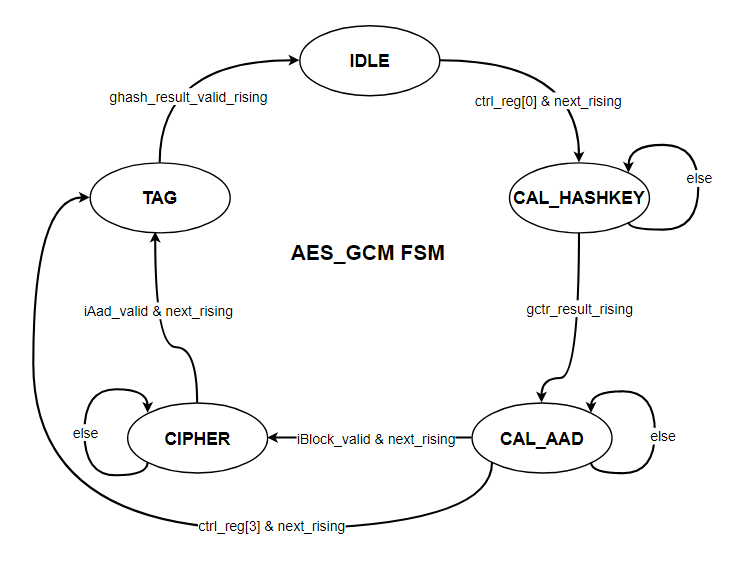
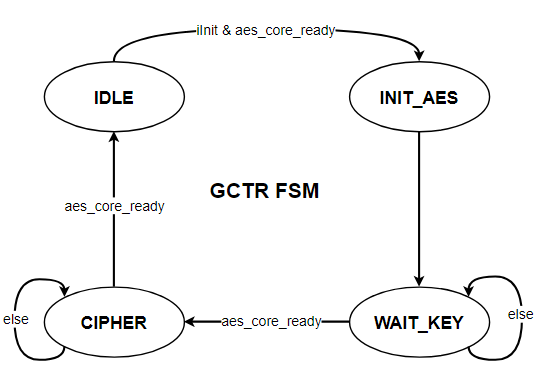




|  |  |
| --- | --- |
| Submodule | File |
| GF\_MUL | gfmul.v |

|  |  |
| --- | --- |
| Name | GHASH\_BLOCK |
| File | ghash\_block.v |

**STATE MACHINEs**



**AES\_GCM FSM**

**IDLE**

gctr\_init = 1’b0

gctr\_hashkey\_compute = 1’b0

gctr\_y0\_compute = 1’b0

ghash\_next = 1’b0

ghash\_input\_signal = 2’b00

ghash\_input\_valid = 1’b0

ghash\_key\_wen = 1’b0

ghash\_key\_valid = 1’b0

ghash result\_wen = 1’b0

y0\_wen = 1’b0

**CAL\_HASHKEY**

if(gctr\_result\_valid)

gctr\_init = 1’b0

else

gctr\_init = 1’b1

gctr\_hashkey\_compute = 1’b1

gctr\_y0\_compute = 1’b0

ghash\_next = 1’b0

ghash\_input\_signal = 2’b00

ghash\_input\_valid = 1’b0

if(gctr\_result\_valid)

gctr\_key\_wen = 1’b1

gahsh\_key\_valid = 1’b1

else

gctr\_key\_wen = 1’b0

gahsh\_key\_valid = 1’b0

ghash\_result\_wen = 1’b0

y0\_wen = 1’b0

**CAL\_AAD**

if(next\_rising & (iBlock\_valid | ctrl\_reg[3]))

gctr\_init = 1’b1

else

gctr\_init = 1’b0

gctr\_hashkey\_compute = 1’b0

gctr\_y0\_compute = 1’b0

ghash\_next = next\_rising

ghash\_input\_signal = 2’b01

ghash\_input\_valid = iAad\_valid

ghash\_key\_wen = 1’b0

ghash\_key\_valid = 1’b1

if(ghash\_result\_valid\_rising)

ghash\_result\_wen = 1’b1

else

ghash\_result\_wen = 1’b0

y0\_wen = 1’b0

**CIPHER**

if(next\_rising)

gctr\_init = 1’b1

else

gctr\_init = 1’b0

gctr\_hashkey\_compute = 1’b1

gctr\_y0\_compute = 1’b0

ghash\_next = gctr\_result\_rising

if(ctrl\_reg[2])

ghash\_input\_signal = 2’b00

else

ghash\_input\_signal = 2’b10

ghash\_input\_valid = gctr\_result\_valid

ghash\_key\_wen = 1’b0

ghash\_key\_valid = 1’b1

if(ghash\_result\_valid\_rising)

ghash\_result\_wen = 1’b1

else

ghash\_result\_wen = 1’b0

y0\_wen = 1’b0

**TAG**

gctr\_init = 1’b0

gctr\_hashkey\_compute = 1’b0

gctr\_y0\_compute = 1’b1

ghash\_next = gctr\_result\_rising

ghash\_input\_signal = 2’b01

ghash\_input\_valid = gctr\_result\_valid

ghash\_key\_wen = 1’b0

ghash\_key\_valid = 1’b1

if(ghash\_result\_valid\_rising)

ghash\_result\_wen = 1’b1

else

ghash\_result\_wen = 1’b0

if(gctr\_result\_rising)

y0\_wen = 1’b1

else

y0\_wen = 1’b0

**GCTR FSM**

**IDLE**

counter\_en = 1’b0

block\_wen = 1’b0

IV\_wen = 1’b0

key\_wen = 1’b0

hashkey\_wen = 1’b0

y0\_wen = 1’b0

aes\_core\_init = 1’b0

aes\_core\_next = 1’b0

gctr\_result\_valid = aes\_core\_output\_valid

**INIT\_AES**

counter\_en = 1’b0

block\_wen = 1’b1

IV\_wen = 1’b1

key\_wen = 1’b1

hashkey\_wen = 1’b1

y0\_wen = 1’b1

aes\_core\_init = 1’b1

aes\_core\_next = 1’b0

gctr\_result\_valid = 1’b0

**WAIT\_KEY**

counter\_en = 1’b0

block\_wen = 1’b0

IV\_wen = 1’b0

key\_wen = 1’b0

hashkey\_wen = 1’b0

y0\_wen = 1’b0

aes\_core\_init = 1’b0

aes\_core\_next = 1’b0

gctr\_result\_valid = 1’b0

**CIPHER**

counter\_en = 1’b1

block\_wen = 1’b0

IV\_wen = 1’b0

key\_wen = 1’b0

hashkey\_wen = 1’b0

y0\_wen = 1’b0

aes\_core\_init = 1’b0

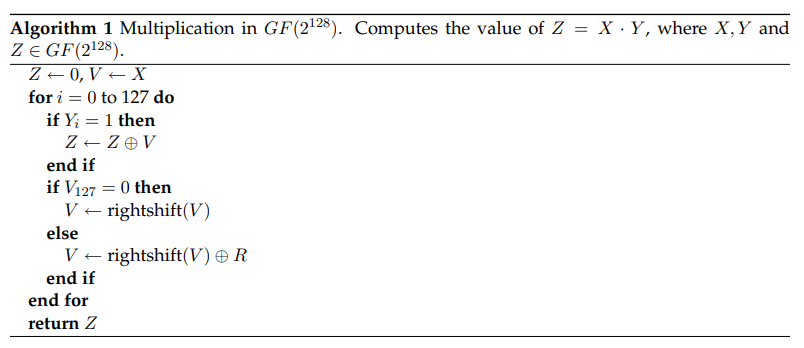
aes\_core\_next = 1’b1

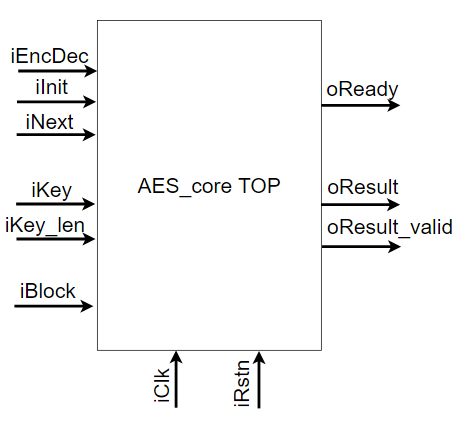
gctr\_result\_valid = 1’b0

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:



URL: [tee-hardware/hardware/teehw/optvsrc/AES at master · uec-hanken/tee-hardware (github.com)](https://github.com/uec-hanken/tee-hardware/tree/master/hardware/teehw/optvsrc/AES)