Stream cipher v2 (AES Inverse S-box)

Project

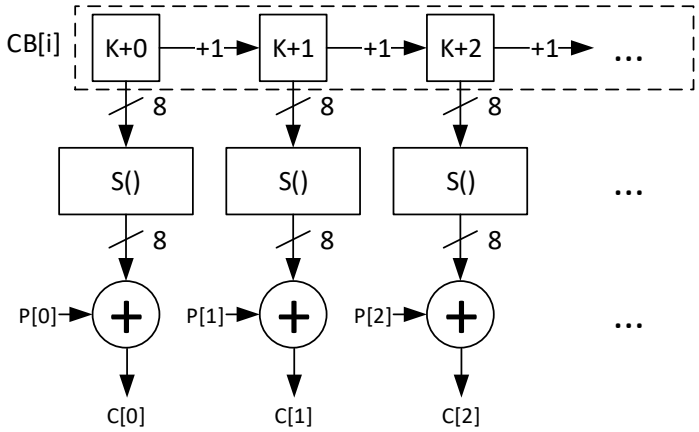
Design and implement an AES S-box based stream cipher supporting both encryption and decryption. The encryption algorithm of cipher is performed by XORing each plaintext byte with an 8-bit value obtained by substituting an 8-bit counter value with the Inverse S-box transformation of the AES algorithm. The 8-bit counter value (Counter Block, CB) has to be initialized with the value of an 8-bit symmetric key, K. The encryption law of the stream cipher can be expressed as it follows:

$$C[i] = P[i] \oplus S(CB[i])$$

where

$C[i]$	is i[th] byte of ciphertext.
$P[i]$	is the i[th] byte of plaintext.
$CB[i]$	is the 8-bit value of the i[th] counter (counter block), for $i = 0,1,2, \ldots$, and it can be represented by the formula $CB[i] = K + i \; mod \; 256$, being K the 8-bit symmetric (encryption/decryption) key. Note mod is the modulo operation.
$S(\quad)$	is the Inverse S-box transformation of AES algorithm, that works over a byte.
$\oplus$	is the XOR operator.

Following figure represents the encryption scheme of the stream cipher.



Additional design specifications

- The module shall have an asynchronous active-low reset port;
- The module interface shall include an input flag to be driven as it follows: 1'b1, when input data byte on the corresponding input port is valid and stable (i.e. it can be used by the internal module logic), 1'b0, otherwise.
- The module interface shall include an output flag to be driven as it follows: 1'b1, when output data byte on the corresponding output port is ready and stable (i.e., external modules can read and use it), 1'b0, otherwise.
- For each arbitrary length plaintext/ciphertext message, the counter block shall start from initialization value K (the 8-bit key). Therefore, the module interface should also include an input flag to signal when a new message begins.

Hints

- The AES Inverse S-box function is largely documented online: implement the LUT version (for faster developing).
  Below it is reported the Inverse S-box of AES algorithm, in hexadecimal format: it works on a byte, using the 4 MSb and the 4 LSb of input byte, respectively, as row and column coordinates to substitute it.

	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
10	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
20	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
30	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
40	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
50	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
60	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
70	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
80	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
90	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
a0	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
b0	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
c0	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
d0	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
e0	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
f0	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

  For example, assuming to apply the Inverse S-box transformation to the byte (hex) 8'h66, the result (hex) is 8'hd3, i.e. the cross between row 60 and column 06: $S(8'h66) = 8'hd3$.

- For decryption focus on following XOR property:
  $$S \oplus S = 0$$
  thus, if $C = P \oplus S$, then $C \oplus S = (P \oplus S) \oplus S = P \oplus S \oplus S = P \oplus (S \oplus S) = P$.