Rotary Substitution Table cipher (Encryption module)

Referent

Email luca.crocetti@phd.unipi.it
Teams l.crocetti@studenti.unipi.it

Project

Design and implement a module that encrypts text characters using the following rotary substitution table:

	$S[1]$	$S[11]$	$S[3]$	$S[9]$	$S[5]$	$S[7]$
$S[0]$	a/A	b/B	c/C	d/D	e/E	f/F
$S[10]$	g/G	h/H	i/I	j/J	k/K	l/L
$S[2]$	m/M	n/N	o/O	p/P	q/Q	r/R
$S[8]$	s/S	t/T	u/U	v/V	w/W	x/X
$S[4]$	y/Y	z/Z	0	1	2	3
$S[6]$	4	5	6	7	8	9

Being $S[0]$, $S[1]$, $S[2]$, …, $S[11]$ the 12 characters of the substitution word S, each letter of the alphabet (case insensitive) and the digits (0 to 9) are substituted with the corresponding pair of S characters in the order row-column; for instance:

- a (or A) -> $S[0]S[1]$
- b (or B)-> $S[0]S[11]$
- u (or U) -> $S[0]S[1]$
- 0 -> $S[4]S[3]$
- 3 -> $S[4]S[7]$
- 9 -> $S[6]S[7]$

Assuming the key K of 12 characters ($K[0]$, $K[1]$, $K[2]$, …, $K[11]$), the substitution word S shall be initialized with the corresponding characters of key K (i.e. $S[0] = K[0]$, $S[1] = K[1]$, …) and used for the first plaintext character substitution, then the S characters shall be circularly shifted on the right (columns characters, $S[1]$, $S[11]$, $S[3]$, $S[9]$, $S[5]$ and $S[7]$) and on the bottom (row characters, $S[0]$, $S[10]$, $S[2]$, $S[8]$, $S[4]$ and $S[6]$), by one position. Thus, for instance:

- first substitution:

		$S[1]$	$S[11]$	$S[3]$	$S[9]$	$S[5]$	$S[7]$
		$K[1]$	$K[11]$	$K[3]$	$K[9]$	$K[5]$	$K[7]$
$S[0]$	$K[0]$	a/A	b/B	c/C	d/D	e/E	f/F
$S[10]$	$K[10]$	g/G	h/H	i/I	j/J	k/K	l/L
$S[2]$	$K[2]$	m/M	n/N	o/O	p/P	q/Q	r/R
$S[8]$	$K[8]$	s/S	t/T	u/U	v/V	w/W	x/X
$S[4]$	$K[4]$	y/Y	z/Z	0	1	2	3
$S[6]$	$K[6]$	4	5	6	7	8	9

- second substitution:

		$S[1]$	$S[11]$	$S[3]$	$S[9]$	$S[5]$	$S[7]$
		$K[7]$	$K[1]$	$K[11]$	$K[3]$	$K[9]$	$K[5]$
$S[0]$	$K[6]$	a/A	b/B	c/C	d/D	e/E	f/F
$S[10]$	$K[0]$	g/G	h/H	i/I	j/J	k/K	l/L

$S[2]$	$K[10]$	m/M	n/N	o/O	p/P	q/Q	r/R
$S[8]$	$K[2]$	s/S	t/T	u/U	v/V	w/W	x/X
$S[4]$	$K[8]$	y/Y	z/Z	0	1	2	3
$S[6]$	$K[4]$	4	5	6	7	8	9

- third substitution:

		$S[1]$	$S[11]$	$S[3]$	$S[9]$	$S[5]$	$S[7]$
		$K[5]$	$K[7]$	$K[1]$	$K[11]$	$K[3]$	$K[9]$
$S[0]$	$K[4]$	a/A	b/B	c/C	d/D	e/E	f/F
$S[10]$	$K[6]$	g/G	h/H	i/I	j/J	k/K	l/L
$S[2]$	$K[0]$	m/M	n/N	o/O	p/P	q/Q	r/R
$S[8]$	$K[10]$	s/S	t/T	u/U	v/V	w/W	x/X
$S[4]$	$K[2]$	y/Y	z/Z	0	1	2	3
$S[6]$	$K[8]$	4	5	6	7	8	9

and so on.

E.g.: assuming the key $K = ABCDEFGHIJKL$ (i.e. $K[0] = A$, $K[1] = B$, $K[2] = C$, …, $K[11] = L$), and the plaintext message $Hello$, then the substitution sequence is:

1. H -> KL

	B	***L***	***D***	***J***	***F***	***H***
A	a/A	b/B	c/C	d/D	e/E	f/F
K	g/G	h/H	i/I	j/J	k/K	l/L
C	m/M	n/N	o/O	p/P	q/Q	r/R
I	s/S	t/T	u/U	v/V	w/W	x/X
E	y/Y	z/Z	0	1	2	3
G	4	5	6	7	8	9

2. e -> GJ

	H	***B***	***L***	***D***	***J***	***F***
G	a/A	b/B	c/C	d/D	e/E	f/F
A	g/G	h/H	i/I	j/J	k/K	l/L
K	m/M	n/N	o/O	p/P	q/Q	r/R
C	s/S	t/T	u/U	v/V	w/W	x/X
I	y/Y	z/Z	0	1	2	3
E	4	5	6	7	8	9

3. l -> GJ

	F	***H***	***B***	***L***	***D***	***J***
E	a/A	b/B	c/C	d/D	e/E	f/F
G	g/G	h/H	i/I	j/J	k/K	l/L
A	m/M	n/N	o/O	p/P	q/Q	r/R
K	s/S	t/T	u/U	v/V	w/W	x/X
C	y/Y	z/Z	0	1	2	3
I	4	5	6	7	8	9

4. l -> ED

	J	***F***	***H***	***B***	***L***	***D***
I	a/A	b/B	c/C	d/D	e/E	f/F
E	g/G	h/H	i/I	j/J	k/K	l/L
G	m/M	n/N	o/O	p/P	q/Q	r/R
A	s/S	t/T	u/U	v/V	w/W	x/X
K	y/Y	z/Z	0	1	2	3
C	4	5	6	7	8	9

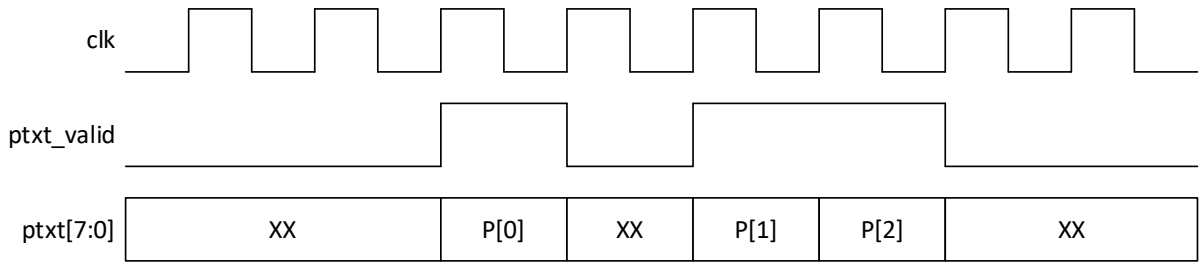
5. o -> EF

	D	***J***	***F***	***H***	***B***	***L***
C	a/A	b/B	c/C	d/D	e/E	f/F
I	g/G	h/H	i/I	j/J	k/K	l/L
E	m/M	n/N	o/O	p/P	q/Q	r/R
G	s/S	t/T	u/U	v/V	w/W	x/X
A	y/Y	z/Z	0	1	2	3
K	4	5	6	7	8	9

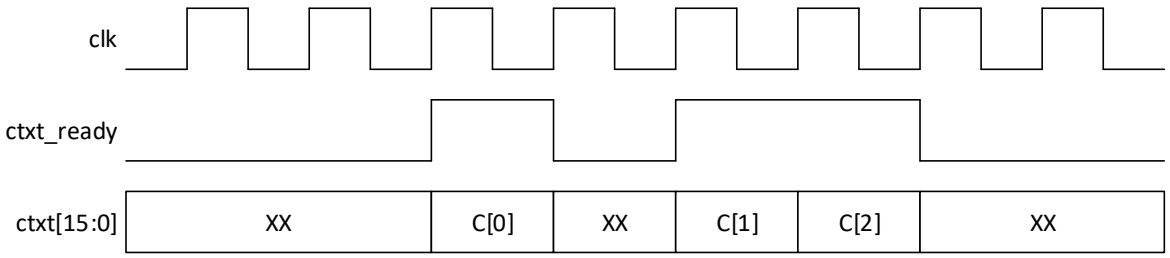
hence the corresponding ciphertext is $KLGJGJEDEF$.

Additional design specifications

- The module shall encrypt one plaintext character per clock cycle;
- The module shall generate one ciphertext symbol per clock cycle (i.e. one pair of substitution characters, over 16 bits, assigning the row substitution character to bits range [15:8] and the column substitution characters to the bits range [7:0]; from the example above, for the first substitution K -> [15:8] and L -> [7:0], for the second substitution G -> [15:8] and F -> [7:0], …);
- The module shall have an asynchronous active-low reset port;
- The key characters can be any 8-bit ASCII code any 8-bit ASCII code of alphabet letters ([a-z] or [A-Z], case insensitive) or digits ([0-9]), but they are not admitted repetition between the key: these conditions shall be checked, and, in case of error (not admitted and/or repeated characters), it must be signalled by asserting a dedicated (output) flag;
- The plaintext character can be any 8-bit ASCII code of alphabet letters ([a-z] or [A-Z], case insensitive) or digits ([0-9]): these conditions shall be checked, and, in case of error (not admitted characters), it must be signalled by asserting a dedicated (output) flag;
- The module shall feature an input port which has to be asserted when providing the plaintext character (*ptxt_valid* port): 1'b1, when input plaintext character is valid and stable, 1'b0, otherwise; the following waveform is expected at input interface of module



- The module shall feature an output port which is asserted when the ciphertext symbol is available at the corresponding output port (*ctxt_ready* port): 1'b1, when output ciphertext symbol is valid and stable, 1'b0, otherwise; this flag shall be kept to logic 1 at most for one clock cycle; the following waveform is expected at the output interface of module



Hints

- It could be suitable to design a module with dedicated logic resources to initialize the substitution table in first instance (by loading the key K – and checking if it is consistent), i.e. that supports a sort of key installation procedure; only then the plaintext can be encrypted. If no key has been installed before, then a dedicated (output) error flag is asserted (for one clock cycle, or until a valid key is installed).