Simplified RC4 stream cipher (decryption module)

Referent

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

Project

Design and implement a simplified version of the RC4 stream cipher for the decryption of ciphertext. The (original) algorithm of RC4 stream cipher for encryption can be found at the following link:

https://en.wikipedia.org/wiki/RC4

As indicated, it essentially consists of two sub-routines: one for the setup of the encryption material (the Key-Scheduling Algorithm, KSA) and one that performs the encryption process (the Pseudo-Random Generation Algorithm, PRGA). The original version of RC4 algorithm foresees to accept as input a symmetric key whose length in bytes (indicated as *keylength*) can vary from 1 up to 256: for the simplified version of RC4 algorithm it is required to fix the key length in bytes to 16 (i.e. 16 bytes, or 128 bits), for which the operation of KSA algorithm

$$j = (\, j + S[i] + key[i \bmod keylength] \,) \bmod 256$$

becomes

$$j = (\, j + S[i] + key[i \bmod 16] \,) \bmod 256$$

In case of decryption, the KSA algorithm is the same, while the decryption process can be implemented by swapping the plaintext bytes with the ciphertext bytes, therefore, the module required to be implemented shall consists of two sub-modules that implement each one of the two following algorithms, respectively, reported as pseudo-C code:

- KSA algorithm:

$for\ i\ =\ 0\ to\ 255\ \{$
    $S[i] = i$
$\}$
$j = 0$
$for\ i\ =\ 0\ to\ 255\ \{$
    $j = (\, j + S[i] + key[i \bmod 16] \,) \bmod 256$
    $swap(S[i], S[j])$
$\}$

- PRGA algorithm (for decryption):

$i = 0$
$j = 0$
$while(1)\{\ \{$
    $i = (i + 1) \bmod 256$
    $j = (\, j + S[i]) \bmod 256$
    $swap(S[i], S[j])$
    $K[l] = S[(S[i] + S[j]) \bmod 256]$
    $P[l] = C[l] \oplus K[l]$
$\}$

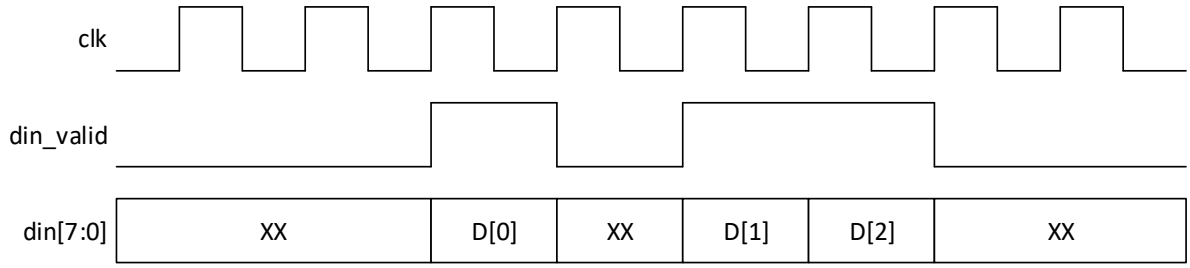
being,

$C[l]$ the l^{th} (input) byte of ciphertext.
$P[l]$ the l^{th} (output) byte of plaintext.
$\oplus$ is the XOR operator.

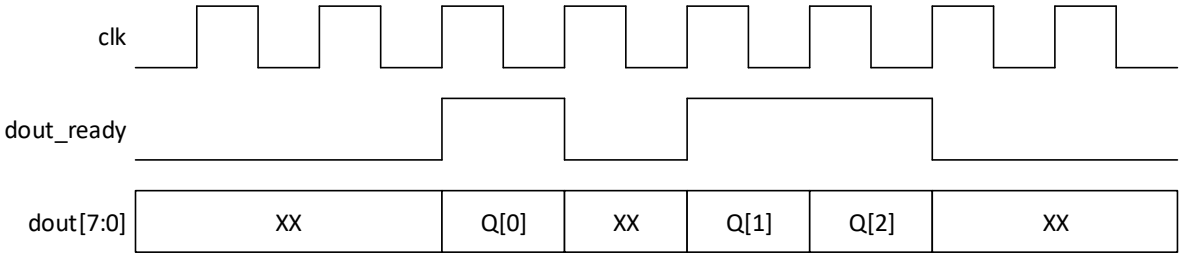
and to be applied to each byte of ciphertext, to generate the corresponding plaintext, one byte per time.

Additional design specifications

- [OPTIONAL] The stream cipher shall decrypt one ciphertext byte per clock cycle;
- The stream cipher shall have an asynchronous active-low reset port;
- The stream cipher shall feature an input port which has to be asserted when providing an input ciphertext byte (*din_valid* port): 1'b1, when input ciphertext byte is valid and stable, 1'b0, otherwise; the following waveform is expected at input interface of stream cipher



- The stream cipher shall feature an output port which is asserted when the generated output plaintext byte is available at the corresponding output port (*dout_ready* port): 1'b1, when output plaintext byte 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 stream cipher



Hints

- The modulo operation (i.e. $A \bmod B$) can be easily implementing in hardware by HDL statements that truncate the n-bit vector A and always return its m least significant bits, being $m = \lceil \log_2(B) \rceil$.
- The computation of $K[l]$ can be performed in advance with respect to the clock cycle in which $C[l]$ is provided as input by means of port din[7:0]: this can be very useful if implementing the optional specifications to support the encryption of one plaintext byte per clock cycle.