Tiny Encryption Algorithm (Encryption module)

Referent

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

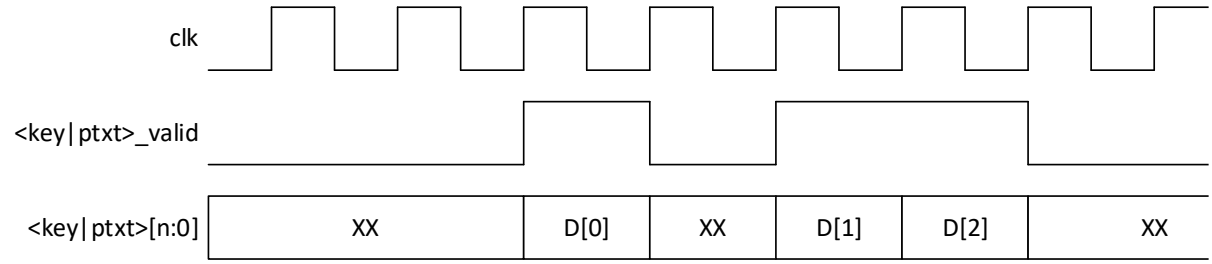
Project

Design a module implementing the decryption function of Tiny Encryption Algorithm (TEA), using as reference the C code reported in section Reference code of online resource https://en.wikipedia.org/wiki/Tiny_Encryption_Algorithm, reported below:

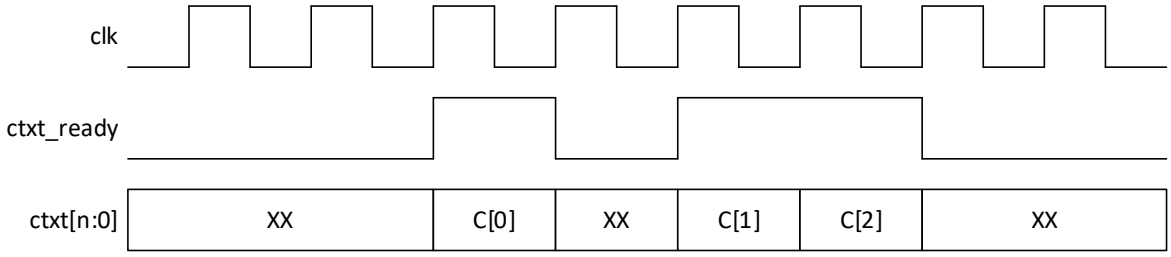
```
void encrypt (uint32_t v[2], const uint32_t k[4]) {
    uint32_t v0=v[0], v1=v[1], sum=0, i;           /* set up */
    uint32_t delta=0x9E3779B9;                     /* a key schedule constant */
    uint32_t k0=k[0], k1=k[1], k2=k[2], k3=k[3];   /* cache key */
    for (i=0; i<32; i++) {                         /* basic cycle start */
        sum += delta;
        v0 += ((v1<<4) + k0) ^ (v1 + sum) ^ ((v1>>5) + k1);
        v1 += ((v0<<4) + k2) ^ (v0 + sum) ^ ((v0>>5) + k3);
    }                                              /* end cycle */
    v[0]=v0; v[1]=v1;
}
```

Additional design specifications

- The module shall have an asynchronous active-low reset port.
- The module shall feature an input port which has to be asserted when providing the input data blocks (key and plaintext) or their parts, *key_valid* and *ptxt_valid* ports, respectively: 1'b1, when input data 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 generated output ciphertext block (or one of its parts) is available at the corresponding output port (*ctxt_ready* port): 1'b1, when output data 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

- No specification on bit width of input and output data ports (key and plaintext/ciphertext): it could fit the data bit width (i.e. 128 bits and 64 bits, respectively), or it could be 32 bits, for instance. In case a bit width lower than nominal bit width of data, please mind that the module should integrate dedicated logic resources (and corresponding input/output ports) to properly load/transfer the data as a sequence of data blocks: for instance 4 32-bit blocks for 128-bit key and 2 32-bit blocks for 64-bit plaintext/ciphertext.
- As general approach, it could be suitable implementing logic that performs 1 of the 64 TEA rounds or, at most, 1 of the 32 rounds pairs (termed cycles).
- ~~For debug, testing and testbench implementation, use hexadecimal format for test vectors. Test vectors can be found at http://tutorialspots.com/test-vectors-tea-3616.html or http://www.cix.co.uk/~klockstone/teavect.htm, or,~~ in addition, the C code at https://en.wikipedia.org/wiki/Tiny_Encryption_Algorithm#cite_note-teapaper-4 could be used to develop a C application that generates test vectors, for both encryption and decryption function.

  Use testvectors uploaded in (shared) project folder: reference_tv.txt