Tiny Encryption Algorithm (encryption)

Project

Design a module implementing the encryption function of Tiny Encryption Algorithm (TEA), using as reference the C code at this link (Wikipedia) and which is 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 interface shall include input flags which has to be driven as it follows: 1'b1, when the corresponding input data (key or plaintext), or its part (e.g., 32-bit data port can be used to provide the key and the plaintext by blocks of 32 bits at time), 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.

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 example, 4 32-bit blocks for 128-bit key and 2 32-bit blocks for 64-bit plaintext/ciphertext.
- 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.