RSA encryption engine

A low-resource, serial RSA engine for Intel FPGA Cyclone V. Mainly suitable for public key operations, implements RSAEP from PKCS #1. Private key operations also work when the key is defined as a pair. This is suboptimal, however, since this engine does not use CRT.

Inspired by: with a few improvements. Unlike the paper, there is no need to precompute the R^2modN whenever a new key is used. A slightly more optimal add/sub structure is also used, which does not require computing the full 2s complement representation of the second term. This is caused by N being the only variable every subtracted and it is being guaranteed to be odd by virtue of the RSA key generation process.

Top level module of the IP block is RSA.sv. This is instantiated inside RSA\_tb.sv for verification and RSA\_wrapper.sv for running on a Cyclone V. Communication is varied out via a register interface. Valid must be asserted for any request to occur.

RSA.sv signals

|  |  |  |  |
| --- | --- | --- | --- |
| Signal | I/O | Width | Function |
| clk | input | 1 | clock |
| reset\_n | input | 1 | synchronous negative edge reset |
| data\_in | input | 8 | data input to pass to internal register |
| data\_out | output | 8 | outputs data inside selected internal register |
| addr | input | 1 | specifies internal register  0=control\_reg, 1= data\_reg |
| valid | input | 1 | Read or write requests only proceed when valid=1. When valid=0, inputs on data\_in, addr and write are ignored. |
| write | input | 1 | Select between read or write operation.  0=read, 1=write |

Registers

control\_reg 0x0

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R/W | R | R | R | R/W | R/W | R/W | R/W | R/W |
| Field | - | - | ready | read\_u | load\_n | load\_e | load\_x | start |

data\_reg 0x1

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R/W | R/W | | | | | | | |
| Field | data\_reg | | | | | | | |

Ready – when asserted, indicates engine is ready to accept new data or start the next operation. If an operation had previously been started, indicates said operation is complete and result is ready.

Load\_n – when set, allows N (modulus) to be inserted/overwritten.

Load\_e – when set, allows E (exponent) to be inserted/overwritten.

Load\_x – when set, allows X (plaintext) to be inserted/overwritten.

Read\_u – when set, begins read back of operation result.

Start – when set, begins RSA operation. Automatically reset by module, is not a status indicator. When operation completes ready is asserted.

Procedure for passing X, E, N

1. Write to ctrl\_reg setting load\_n, load\_e or load\_x bits. Must not be set simultaneously.

2. On the next cycle or any cycle after, write first (least significant) word (8 bits) of X, E or N.

3. Further words can be written on any successive cycle.

Procedure for reading result

1. Write to ctrl\_reg setting read\_u bits. load\_n, load\_e or load\_x must not be set simultaneously.

2. Wait 4 cycles.

3. Read data\_reg register. Contains first (least significant) word (8 bits) of result.

4. Wait one cycle.

5. Read data\_reg for next word.

6. Repeat, leaving one cycle between reads.

Simulation

Simulate used Modelsim Intel FPGA edition or Questa Intel FPGA edition. Modify top two lines of to /Sim/modelsim/sim.do