**Implementation of the RSA cryptosystem on FPGA**

The goal of the project is to implement the RSA cryptosystem on FPGA. Cryptosystems are widely used in security systems, and in most of the cases implementing these are computationally expensive. The RSA cryptosystem is one of such systems, and the core computation in this system is modular exponentiation of large numbers and large modulus. This project is an efficient hardware implementation of modular exponentiation which uses a technique called Montgomery multiplication.

specifications and implementation of modules

* **mont\_mult\_modif**

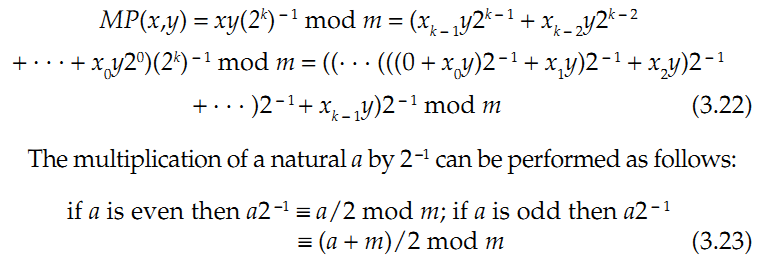
- Computes Montgomery product

Testbench : *modif\_mult\_test*

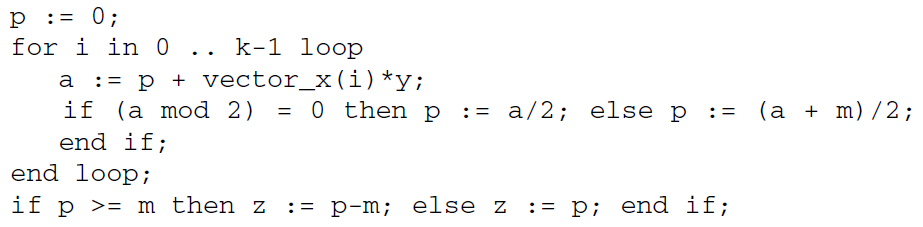
*Inputs* :

*Outputs* : , done signal

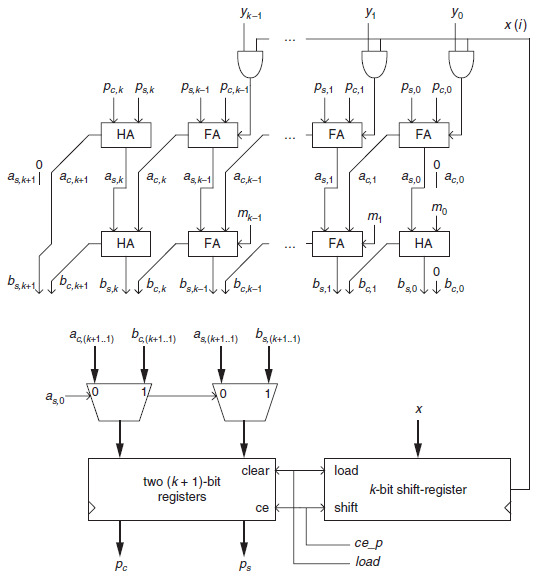
***Mathematical Explanation***

******

***Pseudocode***

******

***Datapath***

****

***State machine***

Default state : *S1* -> jumps to S2 when start signal is raised

*S2* ->

raise load signal

value of x is loaded into shift register

parallel registers are cleared

go to S3

*S3* ->

a counter counts up to the bit length of x (192) and during each clock cycle, the shift register is shifted, and the second, third lines in the above pseudocode are performed. Once the counter reaches 192, go to S4

*S4* ->

A timer is used to down count to 0, during which the final line of the pseudocode is performed. (This is to allow for an alternative subtraction implementation replacing the one which I have used – which will be synthesized using subtraction macros). After timer reaches zero, raise done signal, goto S0

*S0* ->

Make sure that the start signal has been pulsed low after the previous start signal

* **mont\_expo** (Top Module)

- Computes the result of modular exponentiation

Test bench : *mont\_expo\_test*

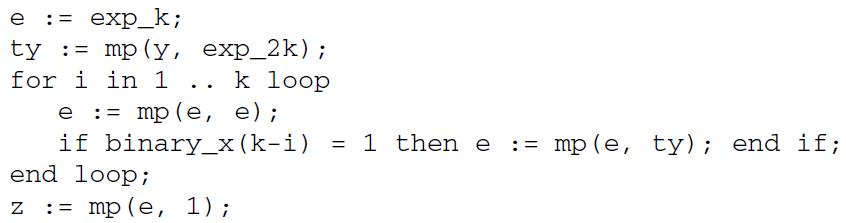
*Inputs* :

*Outputs* : , done signal

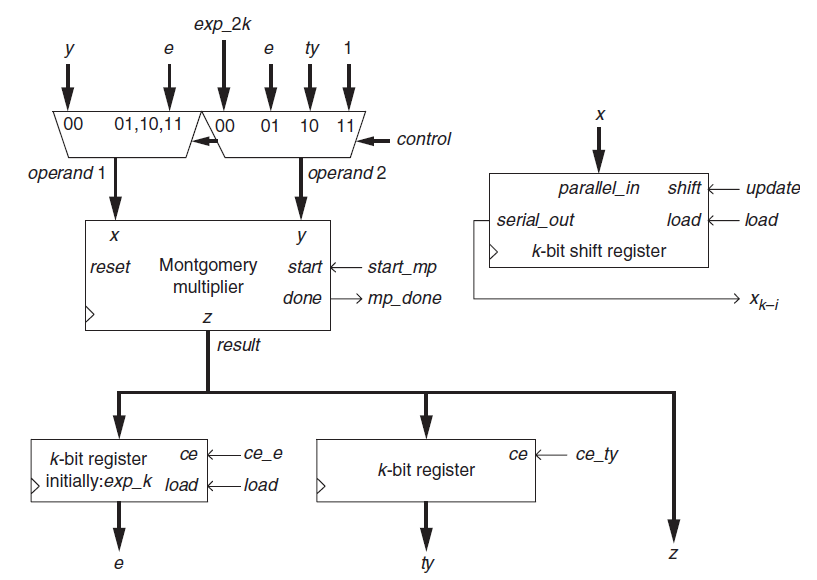
***Mathematical Explanation***

***C:\Users\work\Desktop\pow.PNG***

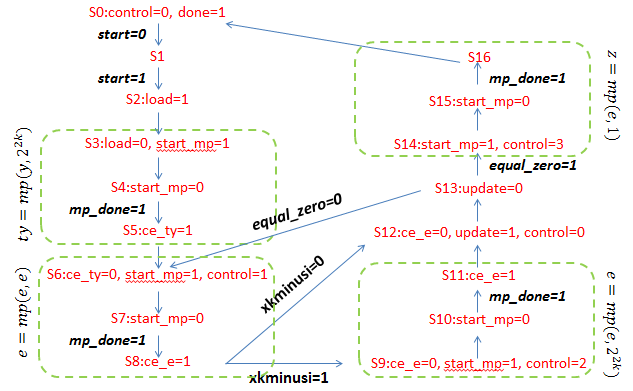
***Pseudocode***

******

***Datapath***

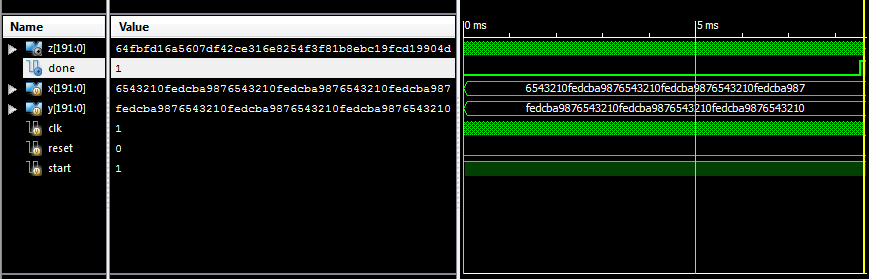
****

***State machine***

The flags that change and the operations performed during each state of the machine are shown in the following figure.

Simulation results

For the case, the run time is at most 100,000 clk cycles



The above figure shows the simulation for typical values of x, y as in the figure and

**References :**

“*Hardware Implementation of Finite-Field Arithmetic*” by Deschamps, Imana and Sutter