Hardware RSA Accelerator

Group 3: Ariel Anders, Timur Balbekov, Neil Forrester

May 10, 2013

Contents

Abstract

Our project is implementing the RSA cryptographic algorithm in Bluespec. The benefits of doing this in hardware are higher performance, reduced power usage and size, and cost. Having reusable IP that implements RSA would allow a device manufacturer to either reduce their power footprint, or skip inclusion of a processor in a device that otherwise would not need one.

More specifically, our objective is to implement the encryption and decryption protocols in RSA for 1024 bit messages on the XUPV5 Bluespec ...speed. (include some speed stuff here)

Chapter 1

Background

CITE WIKIPEDIA!! Assume the readers of your final report are not intimately familiar with your project domain. Provide enough high level background about the domain or algorithms involved for the reader to get an idea of the scope of the project and understand significant design choices described later in the document. Include the benefits of using an FPGA for this application.

RSA is an encryption protocol that involves a public and private key. As indignant of its name, the public key is known to everyone and is used to encrypt messages, which are called "ciphertext". The ciphertext can then be decrypted using the private key. Our implementation assumes the public and private keys are generated prior to encryption through with the FPGA.

1.1 RSA Algorithm

1.1.1 Definition of Variables for the RSA Algorithm

Public Key (n, e) A public key consists of the modulus n and encryption exponent e

Private Key (n, d) A private key consists of the same modulus n and the private decryption exponent d

Message (m) The plain text message converted into an integer m, such that $0 \leq m \leq n$

Ciphertext (c) The plain text message encrypted using the public key

1.1.2 Encryption

The ciphertext is generated by encrypting the plain text message using the public key:

$$c \equiv m^e \ (mod\ n) \tag{1.1}$$

1.1.3 Decryption

The plaintext message is generated by decrypting the ciphertext message using the private key:

$$m \equiv c^d \ (mod\ n) \tag{1.2}$$

1.2 RSA Algorithm for Hardware Implementation

Using a naive approach to implement this algorithm would clearly run quickly out of space. Instead, we have selected two algorithms that blablabla good space yada yada yada Copied algorithms over... still needs to be cleaned up

1.2.1 Modular Exponentiation

This module will employ the Right-to-left binary algorithm, which we believe is a good compromise between speed, memory usage, and complexity. The goal of the algorithm is to calculate $b^e \bmod m$ for very large values of b, e, and m. If the bits of e are $e_1, e_2 \ldots e_n$:

$$e = \sum_{i=0}^{n} e_i 2^i \tag{1.3}$$

then:

$$b^e = \prod_{i=0}^{n} e_i b^{(2^i)} \tag{1.4}$$

and since:

$$a * b \bmod m = (a \bmod m) * (b \bmod m) \bmod m \tag{1.5}$$

then every intermediate result can be taken modulo m to keep the size of intermediate results manageable. Therefore, the following algorithm will compute $b^e \bmod m$ in a reasonable amount of time and memory:

b, e, and m are the inputs to the algorithm.
$c \leftarrow 1$
while $e > 0$ **do**
  if $e \bmod 2 = 1$ **then**
    $c \leftarrow c * b \bmod m$
  end if
  $b \leftarrow b * b \bmod m$
  $e \leftarrow \lfloor e/2 \rfloor$
end while
c is the result of the algorithm.

This very naturally suggests a circular pipeline in hardware. If parallelism is desired, then multiple circular pipelines may be put in parallel, with some

logic at the front and back to manage handing out jobs to different circular pipelines, and collecting the results. The only remaining problem is performing multiplication, modulo, and bit shifting which is implemented through the interleaved modular multiplication algorithm which requires bit shifts, additions, subtractions, and bitwise comparisons and does not take up excessive area.

1.2.2 Interleaved Modulus Multiplication

N is the size of the numbers, in bits. For example, $N = 1024$. Also, x_i is the ith bit of x.

x, y, and m are the inputs to the algorithm.
$p \leftarrow 0$
$i \leftarrow N - 1$
while $i \geq 0$ **do**
    $p \leftarrow p * 2$
    if $x_i = 1$ **then**
        $p \leftarrow p + y$
    end if
    if $p \geq m$ **then**
        $p \leftarrow p - m$
    end if
    if $p \geq m$ **then**
        $p \leftarrow p - m$
    end if
    $i \leftarrow i - 1$
end while
p is the result of the algorithm.

High-Level Design and Test Plan

Describe your system at a high level. What components are running in software, what in hardware, and how do they interact? What are the inputs and outputs to the system? How is the system used? How do you test the system? Include system diagrams.

Microarchitectural Description

Describe the microarchitecture of your project. What are the component modules of your design and how do they interact? What significant design decisions led you to your microarchitecture?

Implementation Evaluation

What challenges or surprises did you face in implementing your design? Did you end up making changes to your original microarchitecture? Did you encounter any problems moving the design from simulation to the FPGA? Describe the results of your implementation. How many lines of Bluespec code did you have to write or modify? What existing IP blocks were you able to make use of in your design (DDR, dividers, etc...). What was the device utilization of your design, the clock frequency, and the high level throughput (frames per second, bits decoded per second, etc...). Did you meet your initial project goals? Why or why not?

Design Exploration

Now that you have a working system, what trade-offs would you be interested in exploring? If you didn't meet your performance targets, why? What are the limiting components in your design? A detailed analysis of the cycle latency and throughput of each component in your design may be appropriate.

Conclusion