Elliptic Curve Group Core
Specification

Author: Homer Xing
homer.xing@gmail.com

Rev. 0.1
February 23, 2012

This page has been intentionally left blank.

Revision History

Rev.	Date	Author	Description
0.1	02/19/2012 02/23/2012	Homer Xing Homer Xing	First Draft 1. correct test bench name for adding two points 2. detailed explain to the speed of scalar multiplication

Contents

1

Introduction

The Elliptic Curve Group core is for computing the addition of two elements in the elliptic curve group, and the addition of c identical elements in the elliptic curve group.

The elliptic curve is super-singular $E: y^2 = x^3 - x + 1$ in affine coordinates defined over a Galois field $GF(3^m), m = 97$, whose irreducible polynomial is $x^{97} + x^{12} + 2$.

The elliptic curve group is the set of solutions (x, y) over $GF(3^m)$ to the equation of E, together with an additional *point at infinity*, denoted O. An element in the elliptic curve group is also called "a point". The elliptic curve group is abelian. The group law is as follows.

Suppose $P_1 = (x_1, y_1), P_2 = (x_2, y_2), P_3 = P_1 + P_2 = (x_3, y_3)$. Then,

$-O = O$
$-P_1 = (x_1, -y_1)$
$P_1 + O = O + P_1 = P_1$
if $P_1 = -P_2$, then $P_3 = O$
if $P_1 = P_2$, then $\lambda = 1/y_1$, $P_3 = \left(x_1 + \lambda^2, -(y_1 + \lambda^3)\right)$
if $P_1 \neq \pm P_2$, then $\lambda = (y_2 - y_1)/(x_2 - x_1)$, $P_3 = (\lambda^2 - x_1 - x_2, y_1 + y_2 - \lambda^3)$

In the following, $P_1 + P_2$ will be also called the addition of (elliptic curve group element) P_1 and P_2. Given a positive integer c, let $c \cdot P_1$ represent the addition of c identical (elliptic curve group element) P_1, i.e. $c \cdot P_1 = P_1 + P_1 + \cdots + P_1$ (c times). Computation of $c \cdot P_1$ proceeds as follows.

1: $A \leftarrow O, B \leftarrow P_1$

2: while $c > 0$ do {

3: if c is odd then $A \leftarrow A + B$

4: $B \leftarrow B + B$

5: $c \leftarrow floor(c/2)$

6: }

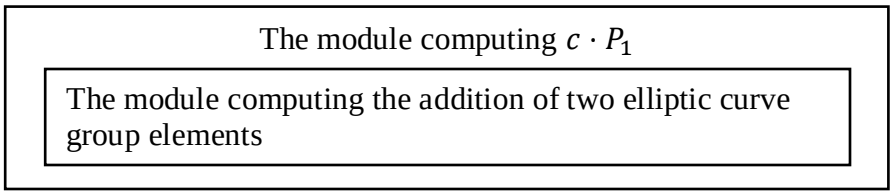
7: return A

For more information about the elliptic curve group, please refer to [1].

2

Architecture

The Elliptic Curve Group core consists of two modules, one computing the addition of two elliptic curve group elements $(P_1 + P_2)$ and the other computing the addition of many identical elliptic curve group elements $(c \cdot P_1)$. The first module is called point_add. The second module is called point_scalar_mult. The second module builds around the first module.



3

Interface

The Elliptic Curve Group core implements the signals shown in the tables below. Two modules are separately listed.

Input signals are synchronous and sampled at the rising edge of the clock. Output signals are driven by flip-flops, and not directly connected to input signals by combinational logic. For signals wider than 1 bit, the range is most significant bit down to least significant bit.

Table 1: Interface signals of point_add

Signal name	Width	In/Out	Description
clk	1	In	clock
reset	1	In	*active high synchronous* reset
x1	194	In	one of input data
y1	194	In	one of input data
zero1	1	In	asserted if (x_1, y_1) is the *point at infinity*
x2	194	In	one of input data
y2	194	In	one of input data
zero2	1	In	asserted if (x_2, y_2) is the *point at infinity*
done	1	Out	The termination signal. It is inactive low after the *reset* signal asserted, and is active high only if the computation is done. When the computation is done , $(x_3, y_3) = (x_1, y_1) + (x_2, y_2)$.
x3	194	Out	Output data. Its value is valid when the *done* signal is asserted.
y3	194	Out	Output data. Its value is valid when the *done* signal is asserted.
zero3	1	Out	Output data. It is active high only if (x_3, y_3) is the *point at infinity*.

Table 2: Interface signals of point_scalar_mult

Signal name	Width	In/Out	Description
clk	1	In	clock
reset	1	In	*active high synchronous* reset
x1	194	In	one of input data
y1	194	In	one of input data
zero1	1	In	asserted if (x_1, y_1) is the *point at infinity*
c	151	In	one of input data
done	1	Out	The termination signal. It is inactive low after the *reset* signal asserted, and is active high only if the computation is done. When the computation is done, $(x_3, y_3) = c \cdot (x_1, y_1)$.
x3	194	Out	Output data. Its value is valid when the *done* signal is asserted.
y3	194	Out	Output data. Its value is valid when the *done* signal is asserted.
zero3	1	Out	Output data. It is active high only if (x_3, y_3) is the *point at infinity*.

4

Timing diagrams

Input timing pattern

Please follow the rule when using any module of the core: When the *reset* signal is asserted, valid input data should be on the input signals and keep valid until the computation termination.

The input data is captured at the moment when the rising edge of the *clk* signal meets the high value of the *reset* signal as shown by the blue arrow in Figure 1.

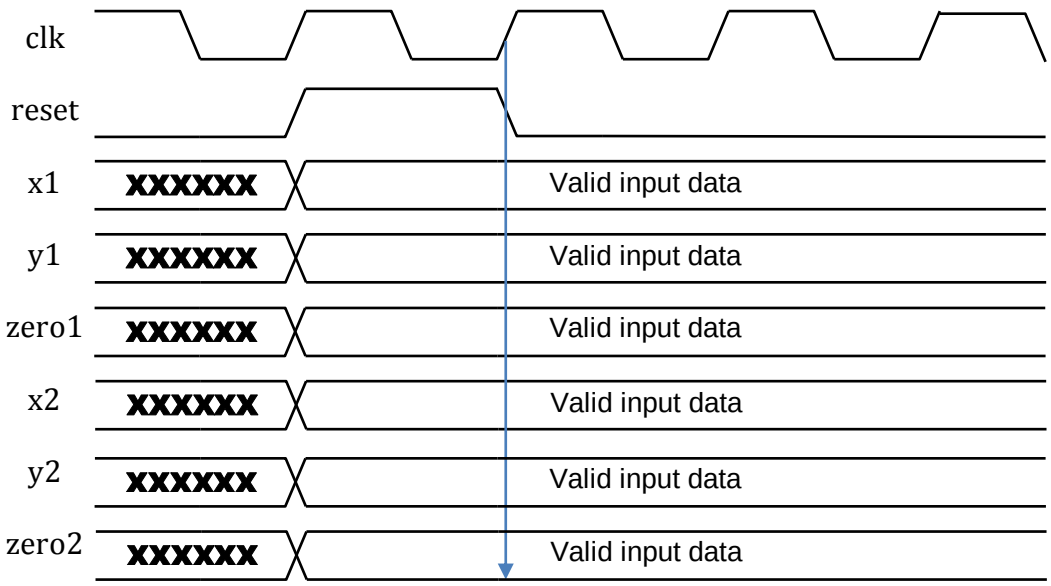


Figure 1: Input timing pattern of the module point_add

Output timing pattern

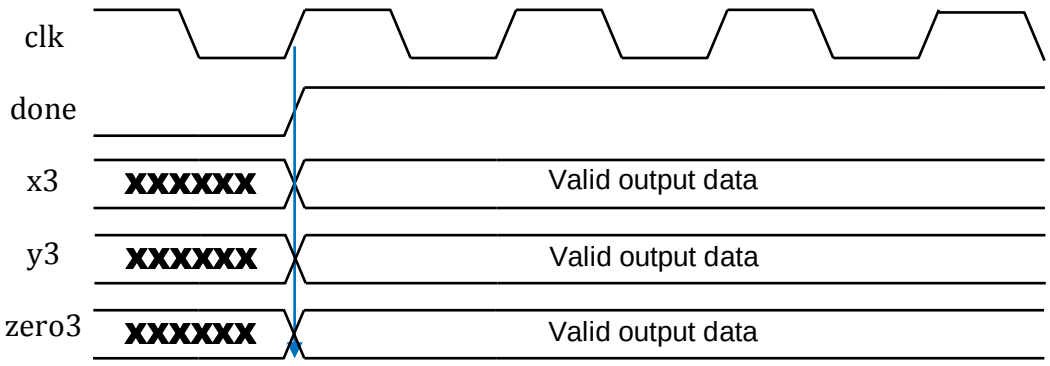


Figure 2: Output timing pattern of the module point_add

5

FPGA Implementation

The core has only been verified on a Xilinx Virtex 4 XC4VLX200-11FF1513 FPGA, synthesized with Xilinx ISE 13.4. For this setup related configuration files is in the directory synthesis/xilinx.

The code is vendor independent in Verilog 2001.

Synthesis results of the module point_add:

- Number of occupied slices: 6,694 (7%)
- Number of 4 input LUTs: 12,099 (6%)
- Number of flip-flops: 6,141 (3%)
- Minimal period: 5.204 ns
- Max achievable frequency: 192 MHz

The module computes $P_1 + P_2$ in 2.7 microseconds if with a 100MHz clock.

Synthesis results of the module point_scalar_mult:

- Number of occupied slices: 7,272 (8%)
- Number of 4 input LUTs: 13,780 (7%)
- Number of flip-flops: 7,451 (4%)
- Minimal period: 6.740 ns
- Max achievable frequency: 148 MHz

The module computes $c \cdot P_1$ in 0.552 milliseconds if with a 100MHz clock. Here c is the order of the generator of the elliptic curve group.

6

Test bench

The file "testbench/simulation.do" is a batch file for ModelSim to compile the HDL files, setup the wave file, and begin function simulation. In order to make it work properly, the working directory of ModelSim must be the directory of "testbench".

The file "testbench/test_point_scalar_mult.v" is the main test bench for the Elliptic Curve Group core. The test bench is self-checked. It feeds input data to the core and compares the correct result with the output of the core. If the output is wrong, the test bench will display an error message. Passing the test bench means both point_add and point_scalar_mult modules work.

The file "testbench/test_point_add.v" is for testing point_add module alone.

7

References

[1] Paulo S.L.M. Barreto, H.Y. Kim, Ben Lynn, and Michael Scott. Efficient Algorithms for Pairing-Based Cryptosystems.