Using MachXO3D ESB to Implement ECC Key Pair Generation

Reference Design

FPGA-RD-02057-1.0

December 2019

Disclaimers

Lattice makes no warranty, representation, or guarantee regarding the accuracy of information contained in this document or the suitability of its products for any particular purpose. All information herein is provided AS IS and with all faults, and all risk associated with such information is entirely with Buyer. Buyer shall not rely on any data and performance specifications or parameters provided herein. Products sold by Lattice have been subject to limited testing and it is the Buyer's responsibility to independently determine the suitability of any products and to test and verify the same. No Lattice products should be used in conjunction with mission- or safety-critical or any other application in which the failure of Lattice's product could create a situation where personal injury, death, severe property or environmental damage may occur. The information provided in this document is proprietary to Lattice Semiconductor, and Lattice reserves the right to make any changes to the information in this document or to any products at any time without notice.

Contents

Figures

Tables

Acronyms in This Document

A list of acronyms used in this document.

Acronym	Definition
ECC	Elliptic Curve Cryptography
ECDSA	Elliptic Curve Digital Signature Algorithm
ECIES	Elliptic Curve Integrated Encryption Scheme
ESB	Embedded Security Block
FPGA	Field Programmable Gate Array
LSE	Lattice Synthesis Engine
TLS	Transport Layer Security
OSC	Oscillator

1. Introduction

Elliptic Curve Cryptography (ECC) is one of the most advanced security algorithms that uses public and private keys to securely transmitted information. ECC enables transport layer security (TLS) to be faster and more scalable on our servers.

ECC has a pair of keys: public key and private key. The public key is a point that lies on the equation of an elliptic curve. The private key can be used to create a digital signature for any piece of data using a digital signature algorithm. Anyone with the public key can verify the authenticity of the signature.

The Embedded Security Block (ESB) is capable of generating a public and private key pair. This function comes in handy for algorithms that require public and private keys such as the Elliptic Curve Digital Signature Algorithm (ECDSA) and the Elliptic Curve Integrated Encryption Scheme (ECIES) protocols. Figure 1.1 shows the high-level overview of the ECC key pair generation algorithm in the ESB.

The first step is to enable the ECC key pair generation engine in the ESB. Wait for some time so that the ESB engine works through the algorithm to generate the output. Once done, public and private keys are ready to be read out.

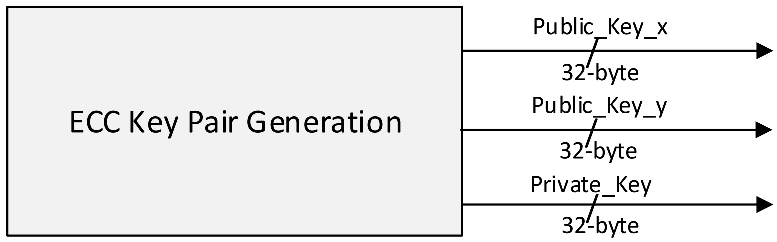

Figure 1.1. High-level Overview of ECC Key Pair Generation

This document describes an ECC key pair generation reference design using the internal hard core of the MachXO3D™ device. The device verifies the generated keys using other two verified ECDSA designs in simulation.

2. Reference Design Overview

This ECC key pair generation reference design shows how to enable the ECC key pair generation engine to generate the public key and private key. The ECDSA design is used for the verification of the generated keys.

2.1. Block Diagram

Figure 2.1 shows the block diagram of the modules. The ECC key pair generation and its connection with the other two modules for the verification can be found in this diagram.

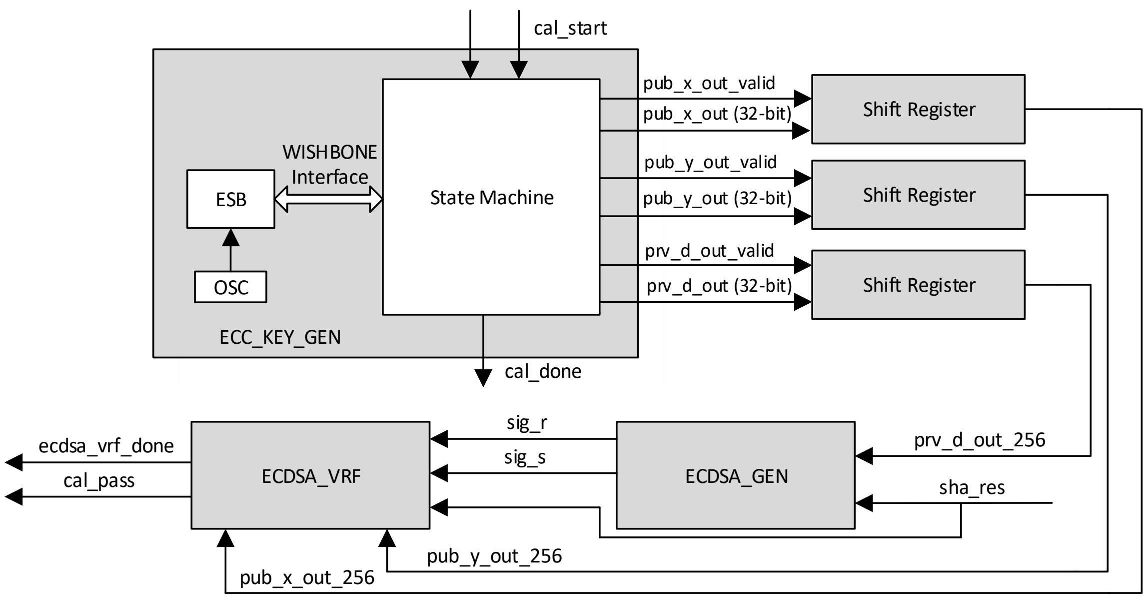

Figure 2.1. Top-Level Block Module and Verification Modules Diagram

2.2. Overview

The ECC key pair generation module consists of three blocks:

- ESB
- Oscillator (OSC)
- State machine

The generation of the ECC keys is executed in the ESB block.

The OSC block has a dedicated OSCESB clock for the ESB block. You can use the OSC output clock or the external clock as the system clock. In this design, the external clock is used for the system clock.

The state machine works as the master of the WISHBONE bus. It configures control registers of ESB and monitors its status register.

The ECDSA generation and verification design is only used for verifying the generated ECC keys. Refer to Using MachXO3D ESB to Implement EDCSA Generation and Verification (FPGA-RD-02053) for more information.

This reference design provides an example for the ECC key pair generation and verification. The features include:

- Randomly generating a pair of public keys and a private key using the ESB engine
- Verifying the generated keys using the ECDSA generation and verification design

3. Functional Description

Initially, the public/private key pair is generated from the ecc_key_gen module and sent to the shift registers to concatenate and get the 256-bit keys.

To verify the generated keys in simulation, the 256-bit private key together with a random data sha_res are fed to the module ecdsa_gen to generate the signature.

Finally, the public keys generated from the first step, the signature generated from the second step and the same data sha_res used in the second step are verified in the module ecdsa_vrf. When the verification is done, the signal ecdsa_vrf_done is asserted high for one clock. At the same time, if cal_pass is high, the verification passes. Otherwise, the verification fails.

4. Design Description

The state machine in the ECC key pair generation design works as the WISHBONE master for the ESB configuration that includes the ESB status registers check and control registers setting. After reset is de-asserted, once the rising edge of cal_start is detected, the state machine issues a series of WISHBONE commands to control the ESB block to generate a pair of public keys and a private key.

4.1. Detailed Input/Output of Design

Figure 4.1 is the I/O diagram of the ECC key pair generation reference design.

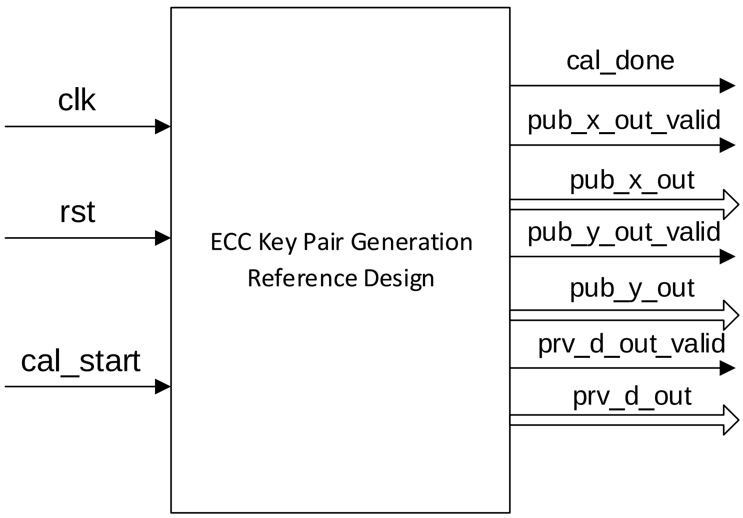

Figure 4.1. I/O Diagram of ECC Key Pair Generation Reference Design

4.2. Pin/Port Description of the Design

Table 4.1 lists the I/O ports of the ECC key pair generation design.

Table 4.1. Pin Descriptions

Ports	Width	Type	Description
clk	1	Input	System clock
rst	1	Input	Asynchronous reset. Active high.
cal_start	1	Input	ECC calculation starts. The rising edge triggers the calculation process.
cal_done	1	Output	ECC calculation done. *1* indicates the calculation is done.
pub_x_out_valid	1	Output	Public key for X direction valid. Active high.
pub_x_out	32	Output	Public key for X direction. 256 bits are generated after shifting eight times of pub_x_out when pub_x_out_valid is 1.
pub_y_out_valid	1	Output	Public key for Y direction valid. Active high.
pub_y_out	32	Output	Public key for Y direction. 256 bits are generated after shifting eight times of pub_y_out when pub_y_out_valid is 1.
prv_d_out_valid	1	Output	Private key valid. Active high.
prv_d_out	32	Output	Private key. 256 bits are generated after shifting eight times of prv_d_out when prv_d_out_valid is 1.

5. ECC Configuration

5.1. ESB Registers for ECC

Table 5.1 lists the registers of the reference design.

Table 5.1. Register Descriptions

Register Type	Register Name	Address	Read/Write	Purpose
Control Register	r0_gp0	18’h2_0020	Read	Check for busy status of ESB 0xB0: READY to get a new command 0xB2: ESB operation done
	ri_ctrl1	18'h2_000c	Write	Enable ESB function 0x0E: Enable ECC Key Pair Gen 0x00: Disable ECC Key Pair Gen
ECC Register	Pub_key_Qx_0	18’h1_F840	Read	Public_Key_x [255:224]
	Pub_key_Qx_1	18’h1_F844	Read	Public_Key_x [223:192]
	Pub_key_Qx_2	18’h1_F848	Read	Public_Key_x [191:160]
	Pub_key_Qx_3	18’h1_F84C	Read	Public_Key_x [159:128]
	Pub_key_Qx_4	18’h1_F850	Read	Public_Key_x [127:96]
	Pub_key_Qx_5	18’h1_F854	Read	Public_Key_x [95:64]
	Pub_key_Qx_6	18’h1_F858	Read	Public_Key_x [63:32]
	Pub_key_Qx_7	18’h1_F85C	Read	Public_Key_x [31:0]
	Pub_key_Qy_0	18’h1_F860	Read	Public_Key_y [255:224]
	Pub_key_Qy_1	18’h1_F864	Read	Public_Key_y [223:192]
	Pub_key_Qy_2	18’h1_F868	Read	Public_Key_y [191:160]
	Pub_key_Qy_3	18’h1_F86C	Read	Public_Key_y [159:128]
	Pub_key_Qy_4	18’h1_F870	Read	Public_Key_y [127:96]
	Pub_key_Qy_5	18’h1_F874	Read	Public_Key_y [95:64]
	Pub_key_Qy_6	18’h1_F878	Read	Public_Key_y [63:32]
	Pub_key_Qy_7	18’h1_F87C	Read	Public_Key_y [31:0]
	Prv_key_0	18’h1_F880	Read	Prv_Key [255:224]
	Prv_key_1	18’h1_F884	Read	Prv _Key [223:192]
	Prv_key_2	18’h1_F888	Read	Prv_Key [191:160]
	Prv_key_3	18’h1_F88C	Read	Prv_Key [159:128]
	Prv_key_4	18’h1_F890	Read	Prv_Key [127:96]
	Prv_key_5	18’h1_F894	Read	Prv_Key [95:64]
	Prv_key_6	18’h1_F898	Read	Prv_Key [63:32]
	Prv_key_7	18’h1_F89C	Read	Prv_Key [31:0]

5.2. ECC Configuration and Generation Flowchart

Figure 5.1 shows the ECC configuration and generation flowchart.

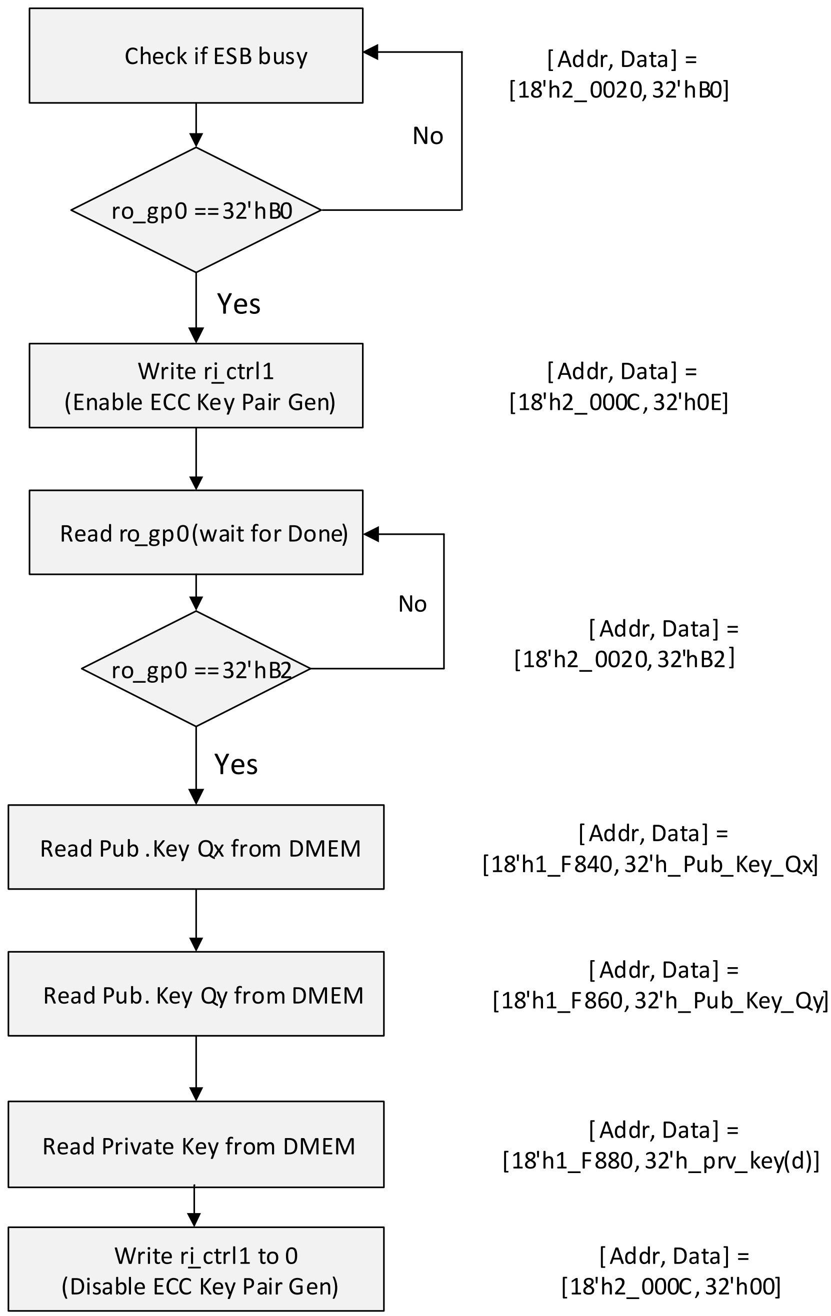

Figure 5.1. ECC Key Pair Generation Process

6. Simulation and Verification

The ECC key pair generation design is verified using the ECDSA generation and verification design. The following figures (Figure 6.1, Figure 6.2, and Figure 6.3) show the waveform for the thumbnail view of the whole simulation process and the detailed timing of the corresponding signals when the ECC Key generation starts and ends.

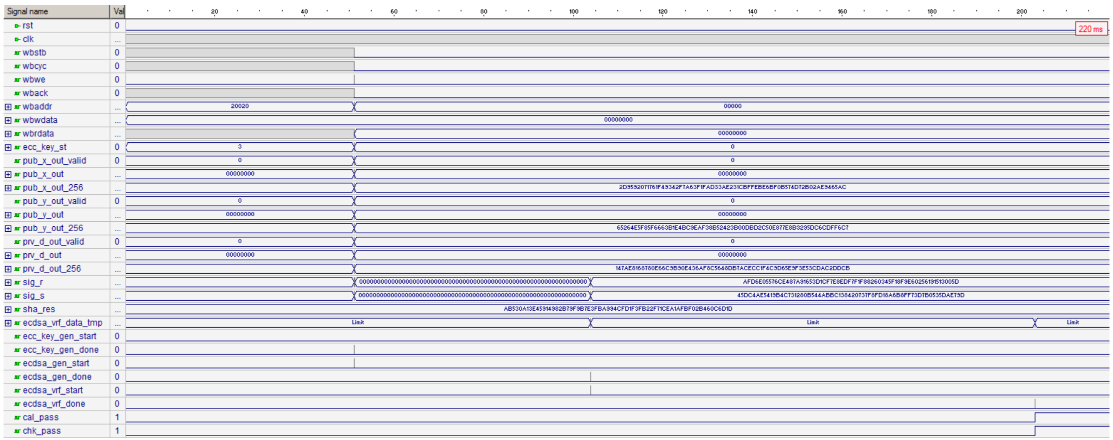

Figure 6.1. Thumbnail View of the Whole Simulation Process

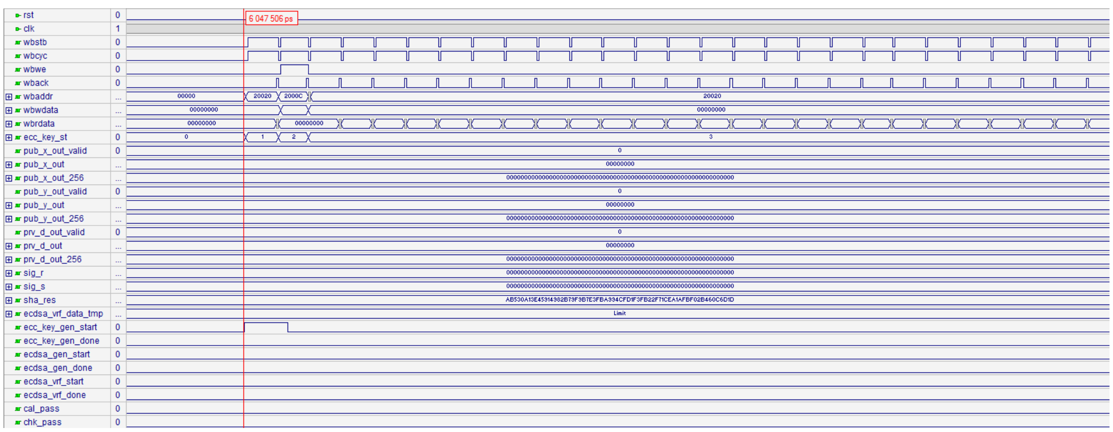

Figure 6.2. WISHBONE Commands that Trigger the Starting of the ECC Key Generation

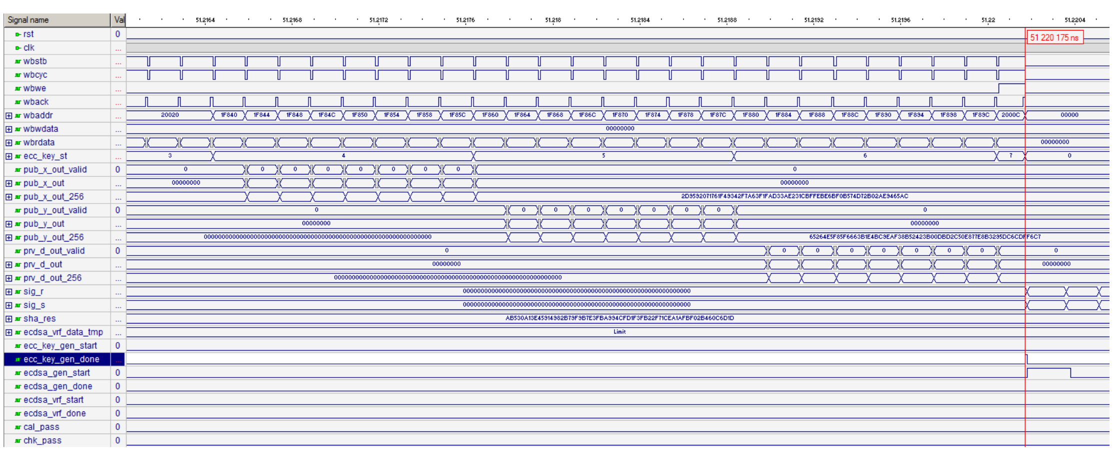

Figure 6.3. ECC Key Generation is Done and ECDSA Generation and Verification Starts

7. Implementation

This reference design is implemented in Verilog HDL using Lattice Diamond® software. The synthesis tool is set to Lattice Synthesis Engine (LSE). When using this design in a different device, density, speed, or grade, performance and utilization may vary.

Table 7.1. Performance and Resource Utilization

Device Family	Language	Utilization	Operating Frequency	ESB Primitive	OSC Primitive	Number of I/O
LCMXO3D-9400HC	Verilog HDL	172 LUTs	>50 MHz	Yes	Yes	103

Note: Performance and utilization characteristics are generated with LCMXO3D-9400HC-6BG484C, using Diamond 3.11 software. The resource utilization is only for the ecc_key_gen module. The ECDSA generation and verification design is not included.

References

- MachXO3D Embedded Security Block (FPGA-TN-02091)
- Using MachXO3D ESB to Implement ECDSA Generation and Verification (FPGA-RD-02053)

Technical Support Assistance

Submit a technical support case through www.latticesemi.com/techsupport.

Revision History

Revision 1.0, December 2019

Section	Change Summary
All	Production release.
Design Description	Changed the following in Table 4.1: • Changed the *pub_x_out_valid* port type from Input to Output, and its description from “Public key for X direction enable” to “Public key for X direction valid”; • Changed the *pub_x_out* port type from Input to Output; • Changed the *pub_ y_out_valid* port type from Input to Output, and its description from “Public key for Y direction enable” to “Public key for Y direction valid”; • Changed the *pub_y_out* port type from Input to Output; • Changed the *prv_d_out_valid* port type from Input to Output, and its description from “Private Key enable” to “Private key valid”.

Revision 0.90, May 2019

Section	Change Summary
All	First preliminary release.

www.latticesemi.com