White Paper

FPGA Design Security Solution Using MAX II Devices

Introduction

SRAM-based FPGAs are volatile devices. They require external memory to store the configuration data that is sent to them at power up. It is possible for the configuration bitstream to be captured during the transmission and used to configure other FPGAs. This form of intellectual property theft can cause revenue loss to the designer.

This document provides a solution to prevent the FPGA designs from being copied. It allows the FPGA design to remain secure even if the configuration bitstream is captured. This is accomplished by disabling the functionality of the user design within the FPGA until handshaking tokens are passed to the FPGA from the MAX® II device. The MAX II devices are selected for generating the handshaking tokens because they are non-volatile and retain their configuration data during power down. In addition, MAX II devices are the most cost-effective CPLDs for such applications. A reference design implementing this solution is also available.

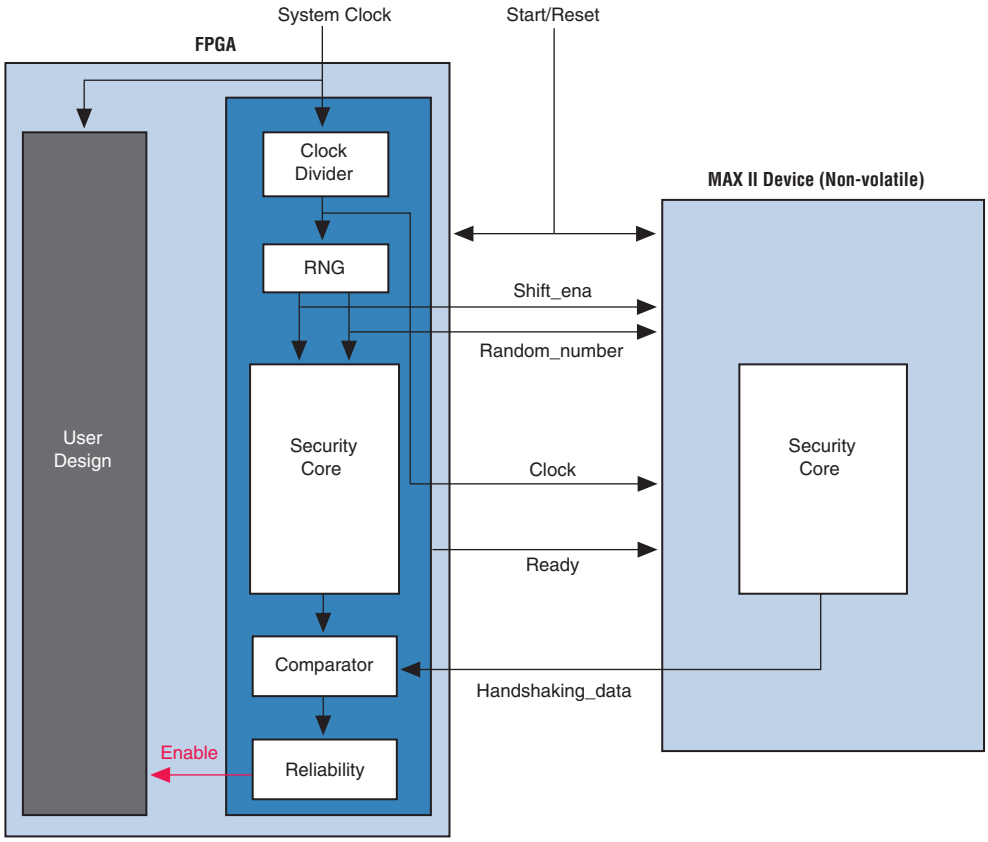
Implementation

The hardware implementation for the FPGA design security solution is shown in Figure 1. The MAX II device generates continuous handshaking tokens that are sent to the FPGA in order to enable the user design. There are 5 signals interfacing between the FPGA and MAX II devices: **clock**, **shift_ena**, **random_number**, **ready**, and **handshaking_data**.

Once the FPGA is configured, it provides continuous clock to the MAX II device. Start/reset signal which is connected to both the FPGA and the MAX II device must be asserted in order to get the system started working. The random number generator (RNG) in the FPGA starts generating the initial counter value for both the FPGA and the MAX II device (the random numbers are sent to the MAX II device only once for every power up or every time start/reset signal is asserted). Once the random numbers are ready, the shift_ena signal goes high and the random numbers are shifted serially to the MAX II device through random_number signal. The ready signal is asserted to indicate the FPGA is ready to receive the handshaking tokens from the MAX II device after the random numbers have been shifted completely to the MAX II device.

The functionality of the user design within the FPGA is disabled after configuration because of the logic low of the Enable signal. The Enable signal is asserted and the user design starting functioning only if the handshaking tokens from the MAX II device match the data generated internally inside the FPGA. When there is a difference between the two data, the Enable signal drops low and the user design is disabled. The method and process used to produce the handshaking tokens within the MAX II device and to generate data within the FPGA device are identical. Without the correct token, the user design within the FPGA device is disabled. This prevents the user design from being copied even if the configuration bitstream is captured.

Figure 1. Hardware Implementation of the FPGA Design Security Solution



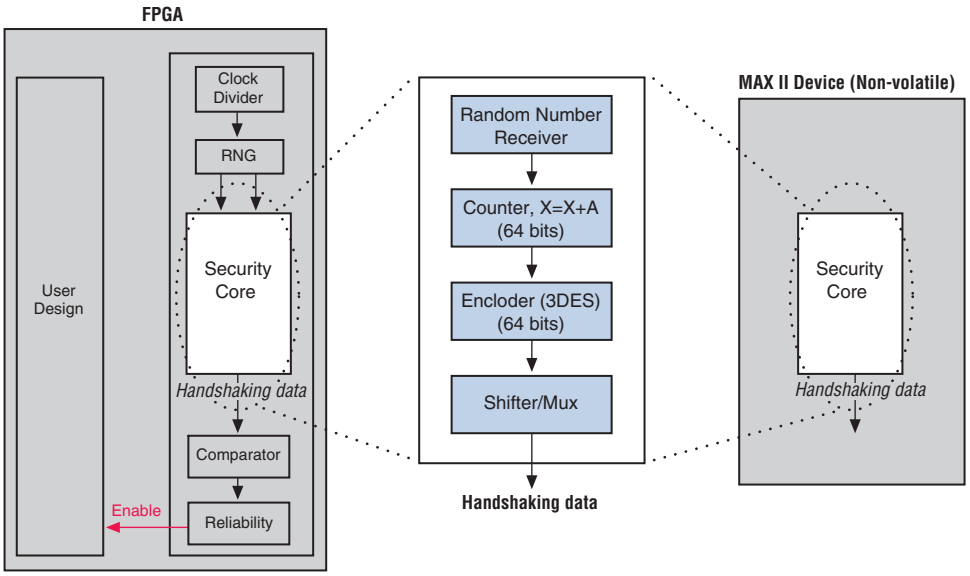
Design Building Blocks

The design security components for the FPGA portion consist of a clock divider, random number generator (RNG), security core, comparator, and reliability, while the design security components for the MAX II device portion are comprised only of the security core as shown in Figure 1.

The security core used in both the FPGA and MAX II devices is the same as shown in Figure 2. It consists of the following parts:

- Random number receiver
- 64-bit counter
- Encoder
- Shifter/mux

Figure 2. Security Core for both FPGA and MAX II Device



Clock Divider

The clock divider inside the FPGA is used to derive a slower clock from the system clock to feed the security core for both the FPGA and the MAX II devices. This is because the security core does not need to run at extremely high frequency. The clock divider is useful for the timing especially when a system runs at high frequency, likewise it can be ignored if a system runs at low frequency.

Random Number Generator (RNG)

The RNG generates the random initial value for the 64-bit counter every time the start/reset signal is asserted. The random numbers are then shifted serially to both security cores for the FPGA and the MAX II device. The reference design uses a 32-bit RNG.

Random Number Receiver

The random number receiver receives the random numbers shifted serially from the RNG and arranges the data in the correct order. The data is fed to the 64-bit counter as the initial value.

64-bit Counter

The 64-bit counter is used to generate the 64-bit data to feed the encoder. It is a simple adder of the form $X=X+A$. X is a 64-bit initial value while A is the increment value of the counter and should be a prime number. The initial value, X comes from the RNG. For the reference design, 32 bits come from RNG and another 32 bits is set by users in the design code. A can be set by users in the design code. The outputs of the counter are fed into the encoder where the data is encrypted. The counter increases its value every time the encoder completes the encryption for previous data.

Encoder

The encoder can be any encryption standards that are difficult to crack. The reference design uses Triple Data Encryption Standard (3DES). The input and output of the 3DES encoder is a 64-bit value, and it takes 48 clock cycles to completely encrypt a 64-bit data.

Shifter/Mux

The shifter/mux takes a limited number of the output bits (16 bits) from the encoder in a specific order, stores them in a register, and shifts them out serially to the comparator while the encoder is preparing the next value.

Comparator

The comparator compares the encoded data (handshaking token) bit by bit from the MAX II device with the encoded data generated internally from the FPGA. The Enable signal is asserted and enables the functionality of the user design if the data from the MAX II device and the FPGA matches. If mismatch occurs, see the *Reliability* section below.

This structure can be duplicated several times to produce more Enable signals to enable different portions of the user design. The duplication can prevent the possibility (the probability is low) of someone hacking the FPGA bitstream to make the Enable signal go high and cause the design security scheme to fail.

Reliability

The reliability helps to handle random bit errors which may cause the system to go down. The reference design allows one data mismatch to occur every ten clock cycles (this is just an example, the user can change the method accordingly to best fit their applications). In other words, the Enable signal will remain high and the system continues working if there is not more than one data mismatch occuring anytime within ten clock cycles. The Enable signal goes low and the user design is disabled when two errors occur within ten clock cycles. The system remains down until the start/reset is asserted to reset the system.

User Design Block

The user design block is the real FPGA design. The Enable signal from the security block should be used to disable the user design block when it is low. In other words, if the comparator encounters any difference between the data from the MAX II device and the FPGA, after taking the reliability into consideration, the functionality of the user design should be disabled.

Figure 3 shows an example of disabling the user design when the Enable signal is low. The FPGA user design shown in Figure 3 has a Clk_en input signal to enable the clocks within the design. The design functions only when the Clk_en signal is high. To implement the design security scheme, the user design is modified slightly (an AND gate is added) so that it is disabled when the Enable signal from the security block is low, as shown in Figure 4.

Figure 3. The FPGA Design Without Security Scheme

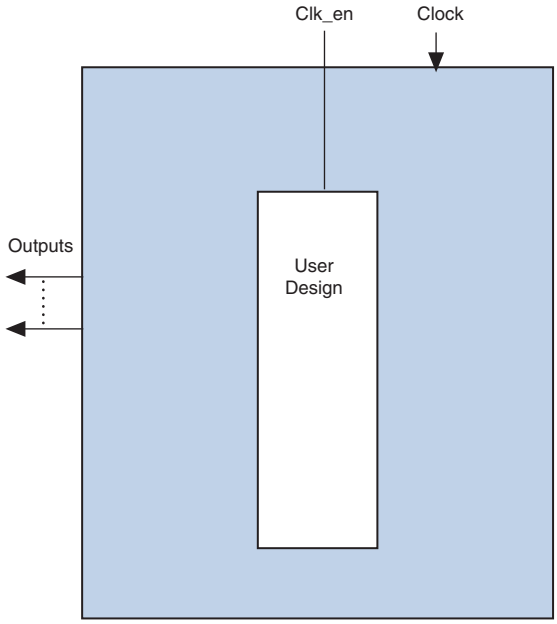
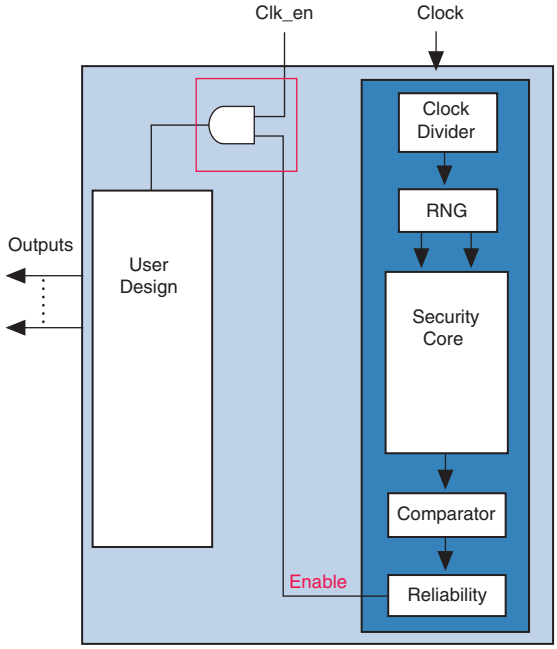


Figure 4. The FPGA Design With Security Scheme



Security of the Solution

The configuration bitstream of an FPGA can be captured when it is being transmitted from an external memory device to the FPGA at power up. The FPGA design can be copied by using the captured bitstream to configure other FPGAs.

With this solution, the FPGA user design only works when the handshaking tokens from the MAX II device matches the data generated internally inside the FPGA. The FPGA design is secure from copying because the duplicated design will not work without the handshaking tokens. The MAX II device used to generate the handshaking tokens is non-volatile and retains its configuration during power-down.

The security of the solution lies on the handshaking tokens generated by the MAX II device. To break this scheme, one needs to either copy the entire bitstream of tokens generated by the MAX II device or figure out the keys used in the encoder to generate the token. Copying the entire bitstream to break this scheme is impossible because the handshaking tokens generated by the MAX II device are different each power-up. This is due to the presence of RNG which generates different numbers feeding the MAX II device at power-up.

It is very dificult to uncover the keys used in the encoder if a proven encryption algorithm is used. In addition, the input data to the encoder is not visible externally and only a portion of the encrypted data is shifted out serially, making it more difficult to conduct plain text attack. A plain text attack is an attack to guess the key by analyzing input and output data of the encoder. Therefore, the FPGA design is protected with this solution.

In order to make the scheme work properly, the system clock feeding the security block should be the same clock feeding the FPGA user design as shown in Figure 1. This is to prevent someone from disabling the security block clock when the Enable signal is high.

Conclusion

This FPGA design security solution protects Altera® FPGA designs from being copied even if the configuration bitstream is captured. It is accomplished by disabling the FPGA user design until handshaking tokens are passed from a MAX II device. The FPGA user design is only enabled if the handshaking tokens match the data generated internally inside the FPGA. With this solution, designers' intellectual property within the FPGA is protected. Please contact Altera for the reference design.

101 Innovation Drive
San Jose, CA 95134
(408) 544-7000
www.altera.com