algotronix
130-10 Calton Road
Edinburgh EH8 8JQ
United Kingdom
Phone: +44 131 556 9242
E-mail: cores@algotronix.com
URL: www.algotronix.com

Algorithm	ZUC
Test Method	Algotronix provided testbench
Performance/Area	Tradeoff via compilation options.
Deliverables	VHDL with Testbench. Targets all modern Intel FPGA families

ZUC (EEA3 and EIA3) IP Core, Intel Edition

Features

- Implements the ETSI/SAGE 128-EEA3 and 128-EIA3 specification based on the ZUC stream cipher.
- Supports 128 bit keys
- Targets all modern FPGA families from Intel
- EEA3 is used for encryption/decryption and EIA3 for authentication
- Supplied as easily customizable portable VHDL to allow customers to conduct their own code review in high-security applications. Compilation options to include only required features and save area

Potential Applications

- 3GPP cellular phone systems

General Description

This IP core implements the ZUC stream cipher and the 128-EEA3 encryption and 128-EIA3 authentication algorithms based on it which are specified for use in 3GPP cellular communications systems.

ZUC is a stream cipher originating in China, it is significantly less computationally complex than the industry standard AES algorithm and provides greater throughput per unit area. A non-pipelined implementation of ZUC with a 32 bit datapath and 128 bit key provides 32 bits of ciphertext per clock where a non-pipelined implementation of AES with a 128 bit datapath provides 128 bits of ciphertext every 10 clocks. Although ZUC has a clear area/performance advantage it should not be assumed that ZUC provides the same level of security as AES. The AES round is more complex than the ZUC operation and AES does multiple rounds of encryption for each block of ciphertext output.

The 128-EIA3 integrity algorithm provides a 32 bit MAC which provides an inherently weaker guarantee of authentication than the significantly longer MACs provided by other integrity algorithms such as AES-CCM or AES-GCM at 128 bits or variants of SHA at several hundred bits.

Use of a ZUC core rather than an AES core is mainly of interest in two scenarios:

a. To implement the EEA3 / EIA3 standards based on ZUC.
b. In applications where high throughput is required and area efficiency is paramount.

The Algotronix ZUC IP Core is supplied as VHDL source code allowing customers to carry out a code review to convince themselves no 'Trojan horse' circuitry has been added to the core which could compromise its security.

The core is supplied with a comprehensive test bench implementing a self-testing version of the synthesizable core which compares its outputs with a behavioral model of ZUC, the behavioral model of ZUC is verified against example vectors in the ZUC standard.

Table 1 shows example implementation statistics for the core in the simple configuration most commonly used for comparison purposes. Algotronix will supply implementation results for the specific core configuration and device required in your application on request.

Table 1: Example Implementation Statistics for 128-EEA3, 'Push Button' flow with Fmax determined by tools. SBoxes in distributed RAM, 128 Bit Key.

Family	Example Device	Fmax (MHz)	ALM	Block RAM	DSP	FF	Throughput (GBit/sec)	Design Tools
Stratix 10	1SG280LN2F-43E1VG	250	992	0	0	617	8	Quartus 19.1

Functional Description

The main functional blocks are shown in Figure 1.

ZUC is a stream cipher, the ZUC algorithm outputs a stream of 32 bit 'random' words derived from a 128 bit key. Encryption is implemented in the 128-EIA3 algorithm by XORing the random data with the input text similar to CTR mode in AES. Encryption and decryption are the same operation so there is no need for control signals or configuration parameters to select between encryption and decryption operations.

Authentication is implemented in the 128-EIA3 algorithm by a fairly complex combining operation over the input message and the keystream coming from ZUC.

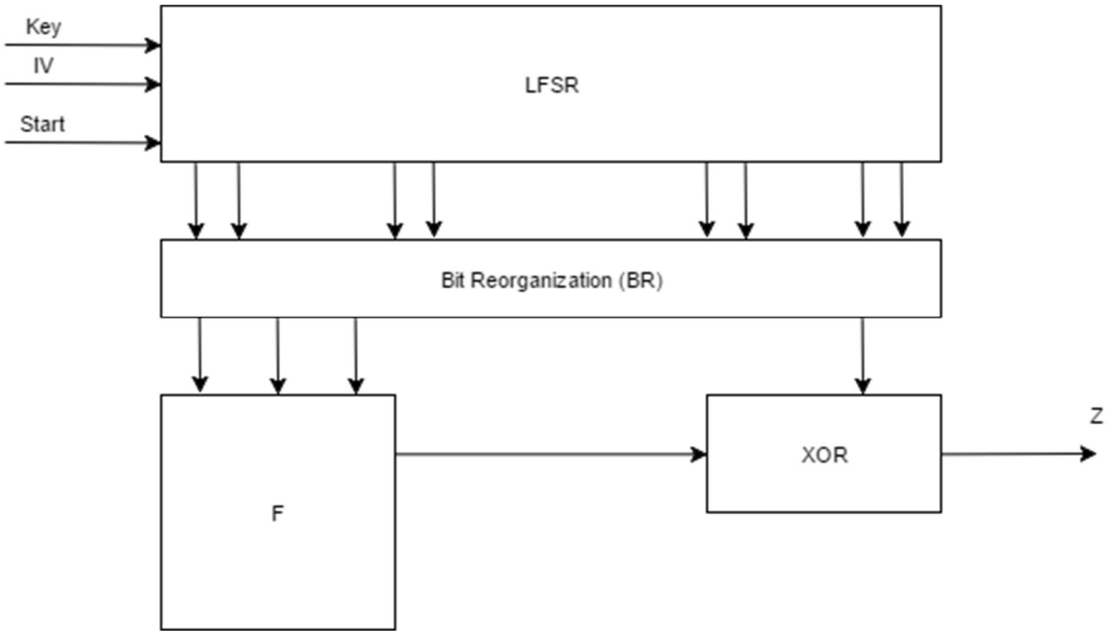

Figure 1, ZUC Core Block Diagram

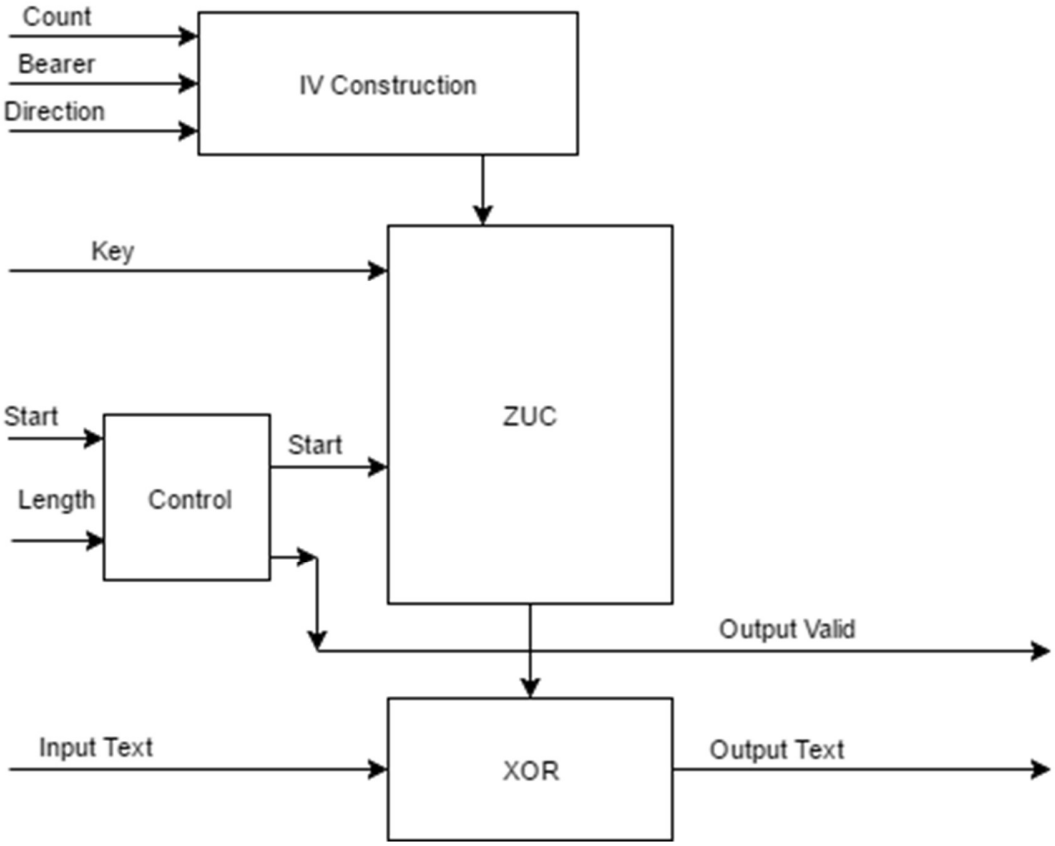

Figure 2, EEA3 Core Block Diagram

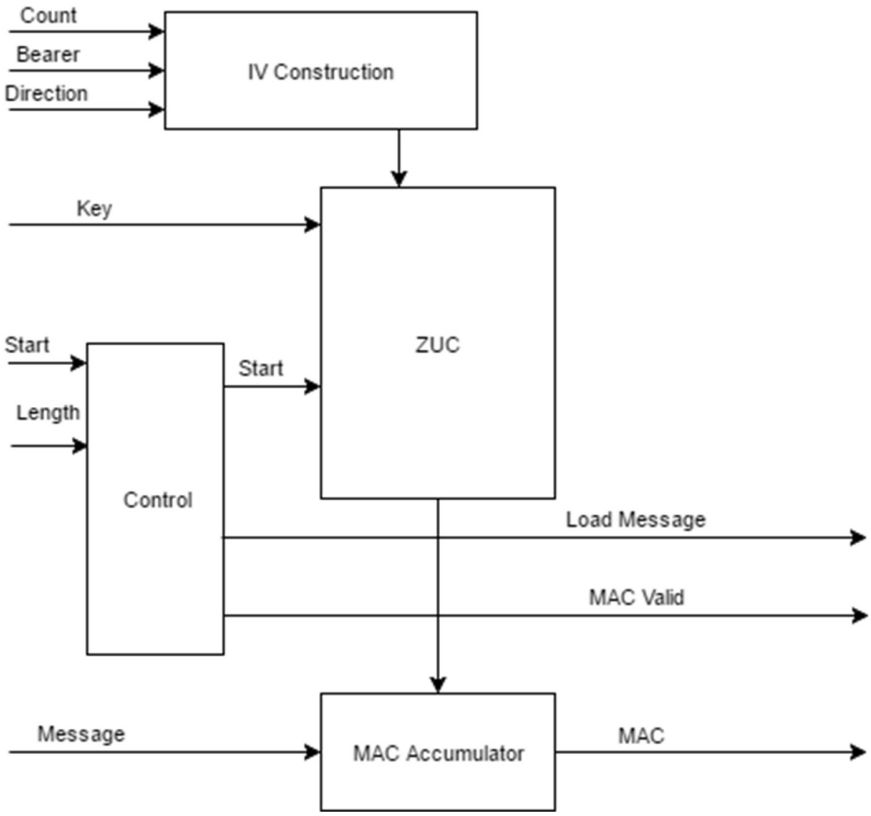

Figure 3, EIA3 Core Block Diagram

Compilation Options

The core can be configured easily using a set of VHDL generic parameters. Normally, it is unnecessary for users to modify the design source code although the code is supplied and they are free to do so if they wish. Algotronix can also customise the core as a service for users with specific requirements which are not met by the standard product.

The following compilation options are specified by editing constant definitions in the zuc_parameters_package file. This is the only file in the ZUC core release which will normally be edited by the user.

- **implement_sboxes_in_ram** – specifies that RAM blocks rather than ALM memory should be used to implement SBoxes. This is often the most efficient option if RAM blocks are available after mapping the remainder of the user design.
- **target_device** – In the Intel Edition version of the core only Intel devices are supported, target device should be set to ALTERA_STRATIX_5 which will support all recent device families with 6 input LUTs.

Core I/O Signals

The core signal I/O have not been fixed to specific FPGA device pins to provide flexibility for interfacing with user logic. Descriptions of all signal I/O for the ZUC, EEA3 and EIA3 cores are provided in Table 2a ,b and c.

Signal	Signal Direction	Description
clock	input	Clock – active on rising edge
reset	input	Reset – active high. A global constant in zuc_package.vhd USE_ASYNCHRONOUS_RESET determines whether this is a synchronous or asynchronous reset.
enable	input	Module clock enable – 0: module is inactive, 1: module operates
start	input	Starts a new encryption operation or block of operations in the chained modes. The control signals mode and do_encrypt are sampled and the parameters fixed for the next operation. The key is assumed to have changed and the keyschedule is recalculated.
iv[127 downto 0]	input	EIA3 count parameter – used to construct IV for ZUC
key[127 downto 0]	input	128 bit key
output_text[31:0]	output	Data output: 32-bit word. There is no corresponding input word, ZUC generates a stream of 32 bit values based on the key and IV.
output_valid	output	Valid flag – high when output_text is valid.

Table 2 a: ZUC Core I/O Signals.

Signal	Signal Direction	Description
clock	input	Clock – active on rising edge
reset	input	Reset – active high. A global constant in zuc_package.vhd USE_ASYNCHRONOUS_RESET determines whether this is a synchronous or asynchronous reset.
enable	input	Module clock enable – 0: module is inactive, 1: module operates
start	input	Starts a new encryption operation or block of operations in the chained modes. The control signals mode and do_encrypt are sampled and the parameters fixed for the next operation. The key is assumed to have changed and the keyschedule is recalculated.
count[31 downto 0]	input	EIA3 count parameter – used to construct IV for ZUC
bearer[4 downto 0]	input	EIA3 bearer parameter – used to construct IV for ZUC
direction	output	EIA3 direction parameter – used to construct IV for ZUC
key[127 downto 0]	input	128 bit key
message_length	input	Length of message in bits.
load_message	output	Indicates core will load the message word present on the message input on the rising edge of the clock.
input_text[31:0]	input	Data input: 32-bit word. Encryption and decryption operations are identical so this could be plaintext or ciphertext.
last_output_word	output	Goes high on the last word of the output message (as determined by message_length input).
output_valid	output	Valid flag – high when output_text is valid.
output_text[31:0]	output	Data output: 32-bit word.

Table 2 b: EEA3 Core I/O Signals.

Signal	Signal Direc-tion	Description
clock	input	Clock – active on rising edge
reset	input	Reset – active high. A global constant in zuc_package.vhd USE_ASYNCHRONOUS_RESET determines whether this is a synchronous or asynchronous reset.
enable	input	Module clock enable – 0: module is inactive, 1: module operates
start	input	Starts a new encryption operation or block of operations in the chained modes. The control signals mode and do_encrypt are sampled and the parameters fixed for the next operation. The key is assumed to have changed and the keyschedule is recalculated.
count[31 downto 0]	input	EIA3 count parameter – used to construct IV for ZUC
bearer[4 downto 0]	input	EIA3 bearer parameter – used to construct IV for ZUC
direction	output	EIA3 direction parameter – used to construct IV for ZUC
key[127 downto 0]	input	128 bit key
message_length	input	Length of message in bits
load_message	output	Indicates core will load the message word present on the message input on the rising edge of the clock
message[31:0]	input	Data input: 32-bit word
output_valid	output	Valid flag – high when MAC is valid.
mac[31:0]	output	Calculated MAC.

Table 2 c: EIA3 Core I/O Signals.

Description of Operation

The input_text, output_text and message buses transfer 32 bit ZUC block values over 32 bit wide buses in a single clock cycle. The input_key bus transfers 128 bits of key information in a single clock. Words are transferred in order starting with the most significant word.

Synchronising between the user design and the core is straightforward. The user controls when the core processing starts using the 'start' signal. When the core samples 'start' as being high on a rising edge of the clock it latches the parameters for a new chain of encryptions and on the following clock cycle loads key data and starts the initialisation phase of ZUC.

Once the initialisation is complete the 128-EEA3 core loads a 32 bit input text word and outputs a 32 bit word on every clock cycle. The user design can use the enable signal to implement flow control if it is not ready to accept the output data – bringing enable low will stop the core processing.

The best way to get an understanding of the timing of the interface signals to the core is to simulate the core using the testbench provided and examine the waveforms on the signals in the table above during operation in the mode of the cipher you wish to use.

Verification Methods

The testbench includes a self-checking configuration of the top-level entity in the VHDL design which uses a behavioral model of the ZUC algorithm to check the results from the synthesisable implementation code. This is implemented using the VHDL facility to provide multiple architecture definitions for a particular entity: the top-level entity in the design has a self-checking and a synthesis architecture defined. The ZUC behavioral model is validated using known-answer examples provided in the ZUC standard and by loop-back testing.

As shown in Figure 4, the self-checking architecture has an identical interface to the synthesisable architecture and instances the synthesisable architecture within itself but also contains behavioral code to capture all input and output signals and check their values against expected values computed using a behavioral model. When errors are detected assertions are triggered and the simulation is stopped with an error message.

This self-checking configuration of the ZUC core can also be instantiated within the user's own simulations. This makes it easy to verify the core operates properly when connected to the user circuitry surrounding the core. In addition, the assertions within the self-checking code will detect and report many situations where the user design is not driving the core correctly simplifying the task of integrating the core with the larger user design easier.

The ZUC testbench supplied with the core also makes use of the self-checking configuration of the core for random testing. The testbench stimulates the self-checking core with a random sequence of test data and the self-checking core takes responsibility for detecting any errors.

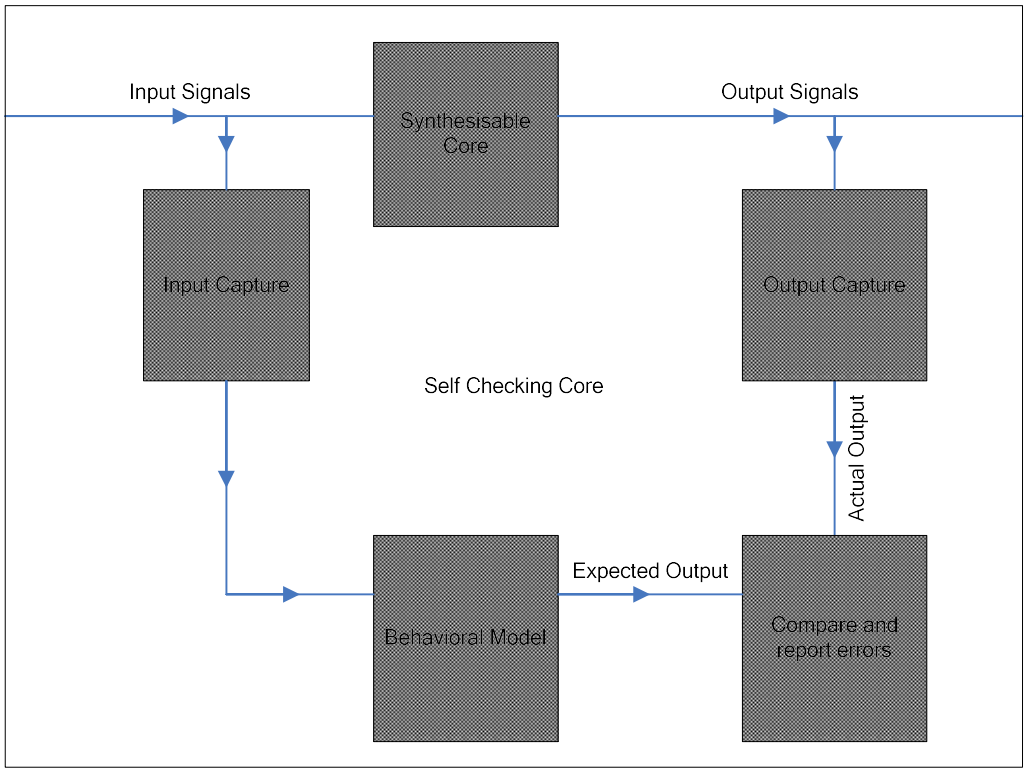

Figure 4, ZUC Self Checking Architecture

Customization Service

Algotronix can offer a cost-effective customization service for this core in order to tune the implementation for easy integration into a larger system. It is also possible to produce variants with significantly higher performance at the expense of increased area and to create optimized variants of the core targeted at specific FPGA families.

Recommended Design Experience

It is recommended that the user is familiar with the VHDL language and with the Intel design flow and simulation tools. The core can also be instantiated inside a wrapper to allow use with a Verilog design flow.

It is also recommended that the user has a background in data security or takes appropriate advice when considering how to implement ZUC in a larger system.

Export Control

Strong encryption technology such as ZUC is the subject of international export regulations. Algotronix is located in the United Kingdom and export of this core is regulated by the UK government.

The core is freely available within the European Union and in addition can be supplied immediately to the following countries: United States, Australia, New Zealand, Canada, Norway, Switzerland, Japan.

Export to other countries requires an export licence. The UK Department of Business, Enterprise and Regulatory Reform publishes information on their website (www.berr.gov.uk) which gives an indication of average export licence processing times for various countries and the percentage of licence requests which are granted. For many countries obtaining an export licence can be done relatively quickly and with only a small amount of paperwork.

It is the the responsibility of the customer to comply with all applicable requirements with respect to re-export of products containing the ZUC technology.

Related Standards

ETSI/SAGE: “Specification of the 3GPP Confidentiality and Integrity Algorithms 128-EEA3 & 128-EIA3. Document 1: 128-EEA3 and 128-EIA3 Specification”, Version 1.6, 1 July 2011.

ETSI/SAGE: “Specification of the 3GPP Confidentiality and Integrity Algorithms 128-EEA3 & 128-EIA3. Document 2: ZUC Specification”, Version 1.6, 28 June 2011.

ETSI/SAGE: “Specification of the 3GPP Confidentiality and Integrity Algorithms 128-EEA3 & 128-EIA3 Document 3: Implementor’s Test Data”, Version 1.1, 4th January 2011.

ETSI/SAGE: “Specification of the 3GPP Confidentiality and Integrity Algorithms 128-EEA3 & 128-EIA3. Document 4: Design and Evaluation Report”, Version 1.3, 18 January 2011.

Algotronix Ltd.
130-10 Calton Road
Edinburgh EH8 8JQ
Phone: +44 131 556 9242
E-mail: cores@algotronix.com

URL: www.algotronix.com