ES1020 - ARC4 Core

Introduction

ES1020 core fully implements the ARC4 stream ciphering algorithm in hardware. For each input character, an output character is nerated. ES1020 ARC4 core allows simultaneous encryption/decryption of multiple independent sessions in a time-multiplexed fashion. Each ARC4 data ciphering session has associated a context. In ARC4, a ciphering context is fully defined by its encryption/decryption key and the internal state of the cipher (256 Bytes). Once the state of the cipher is initialized by the key, the key is no longer necessary for further operation.

ARC4 core includes a direct interface to a context memory. This allows very fast context switching between different sessions. The interface to context memory is generic enough to match most common types of compiled memory.

ARC4 operation is identical for ciphering and deciphering so a single command cipher/decipher is provided for that purpose.

Features

- ✓ Supports ciphering of multiple sessions in a time multiplexed fashion (maximum number of sessions is parametrically defined at compile time).
- ✓ Direct interface to context (state/key) memory.
- ✓ CPU i/f provided through the core to initialize and access context memory, simplifies memory i/f logic.
- ✓ 256 cycles required to initialize each session context.
- ✓ After context has been initialized, the core sustains a throughput of 8 bits every 4 cycles (e.g. 500 Mbps throughput at 250MHz).
- ✓ Simple external interface
- ✓ Zero overhead context switching.
- ✓ Available in ASIC and FPGA technologies.
- ✓ Minimal gate count implementation.

Applications

- Secure personal and corporate communications
  - Wireless LAN's (WLAN's)
  - Wireless Access Points
- Secure electronic transactions
  - SSL web server acceleration
  - eComerce

Pin Description

Name	I/O	Width	Description
Clk	Input	1	Positive edge active clock
rst_n	Input	1	Active low asynchronous reset
Cmd	Input	2	Commands supported are: 00 – Init context 01 – Cipher/Decipher 10 – Read Mem 11 – Write Mem
cmdVld	Input	1	Validates cmd signal
cmdBsy	Output	1	Indicates current cmd is in progress and is not ready to accept a new command
cmdPars	Input	PARS_WIDTH	Used to provide extra information for each command
Din	Input	8	Used to feed data to encrypt/decrypt and CPU data to write into context memory
dinVld	Input	1	Validates din
dinBsy	Output	1	Used to flow control din stream of data. When high, new data won't be accepted by the core.
Dout	Output	8	Used to extract encrypted/decrypted data out of the core and data read from memory by the CPU
doutVld	Output	1	Validates dout
doutBsy	Input	1	Used to flow control dout. It eventually flow controls the input.

Table 1. Interface to core

Name	I/O	Width	Description
memAddr	Output	MADDR_WIDTH	Address that ARC4 core is accessing from context memory
memReq	Output	1	Indicates a valid read/write cycle initiated by ARC4 on context memory
memWrite	Output	1	Write access when '1', Read access when '0'
memWdata	Output	8	ARC4 Data to write into context memory
memRdata	Input	8	Data received from memory
memRdataVld	Input	1	Validates memRdata
memBsy	Input	1	Indicates that memory is not available to receive a read or write request. This signal can be tied to '0' if context memory is not shared with any other device.

Table 2. Interface core-memory

Functional Description

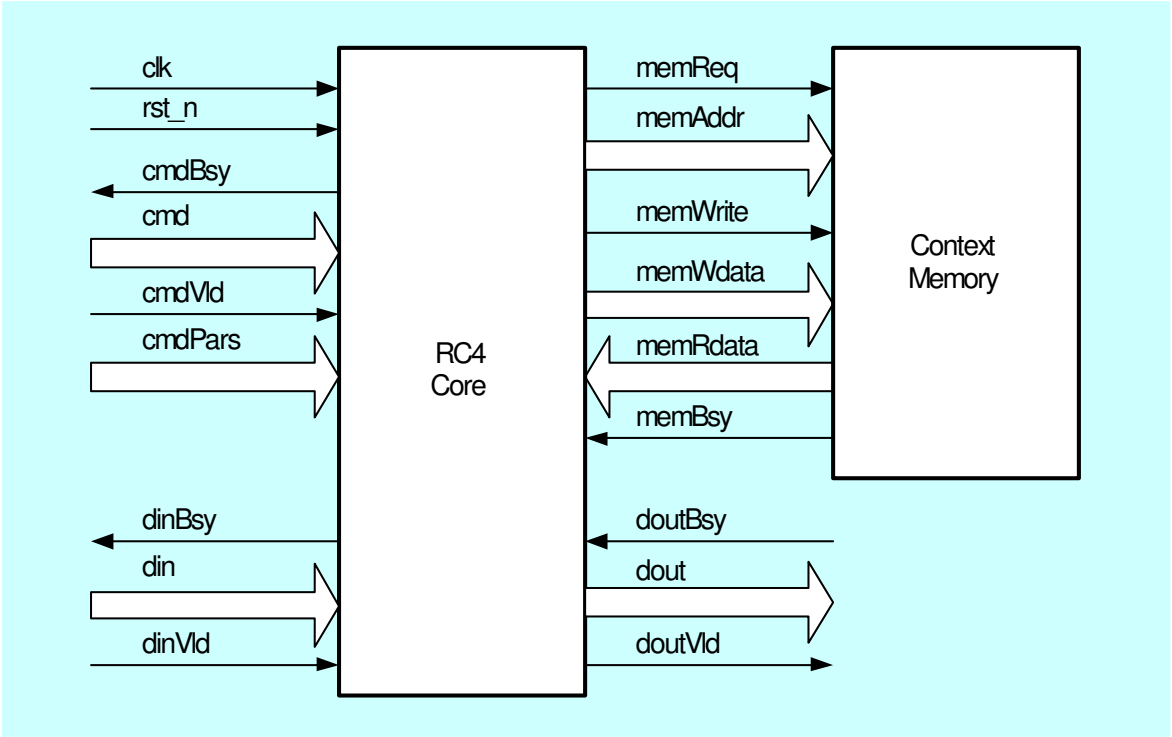

Figure 1. ES1020 Symbol

To perform an encryption operation with ES1020 ARC4 core, the following sequence of operations must be performed.

1. Key storage on context memory. Key is stored in memory in the form {N = key Length, Key[1..N]}. CPU writes this information into context memory using CPU_WR command (recommended) or through a direct i/f to context memory (if user provides it).

2. State initialization. Before any ciphering / deciphering operation is performed, the internal cipher state must be initialized. This step mangles key information with the internal state of the cipher (aprox 256 cycles). cmdBsy = '0' will signal the readiness of the core to proceed with further commands. Init command requires two extra parameters that allow the core to identify where the key is located in context memory (keyPtr) and where the state of the cipher will be stored (slot).

3. Cipher/Decipher operation. Any number of bytes can be cipher/decipher. They are fed into the core one byte at a time. CIPHER/DECIPHER command initiates this operation. One extra parameter is provided with the command: the state slot number. This number will be used to generate the address where the state of the session is stored by multiplying it by 256 (for example Slot = 4 will use the state stored in address 1024 to 1279)

Interface Description

Figure 2 shows the timing diagram of the INIT command. Two parameters are provided with the init command

- KP: Key Pointer. Byte address of the beginning of the key in context memory. Key is stored as a keyLength byte followed by as many bytes as KeyLength specifies.
- S: State Slot number. A State Slot consists of 256 bytes. This number is multiplied by 256 to generate the address in context memory where the state of the session is stored. ARC4 operation reads and modifies state context.

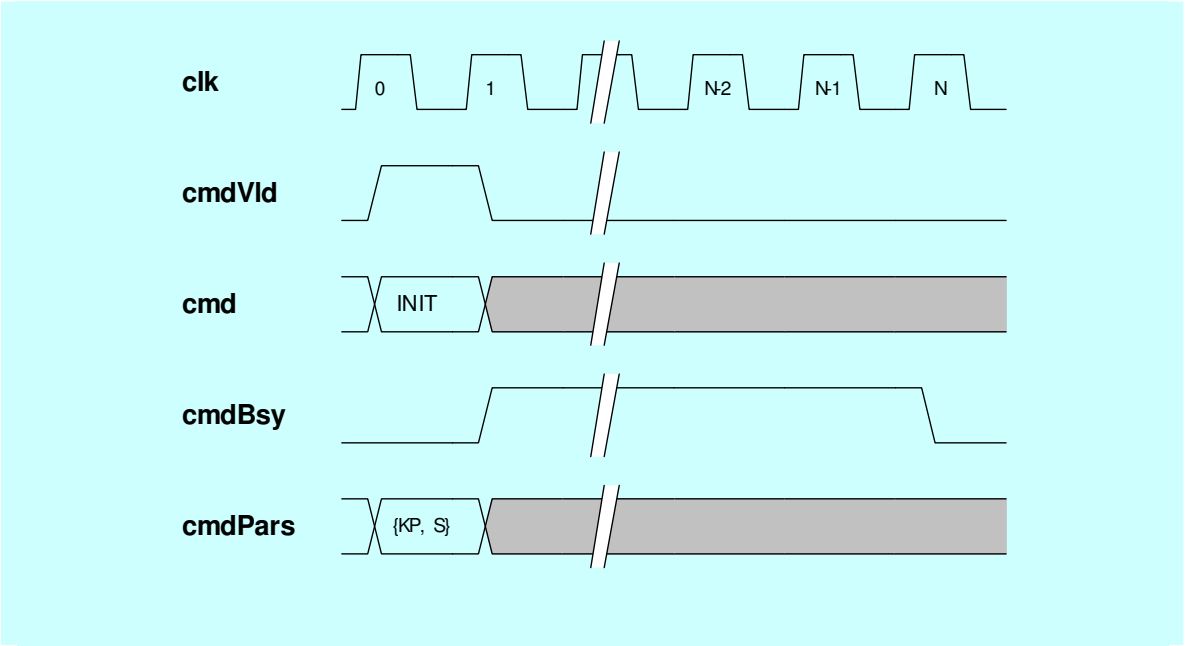

Figure 2. INIT command Timing Diagram

Figure 3 shows the timing diagram of the Crypt/Decrypt command. Any number of bytes can be encrypted / decrypted through this command. Time Division Multiplexing of the core is supported by supporting and keeping multiple contexts in context memory. It is up-to the external logic to break the data in packets of limited size, if needed, for a given application.

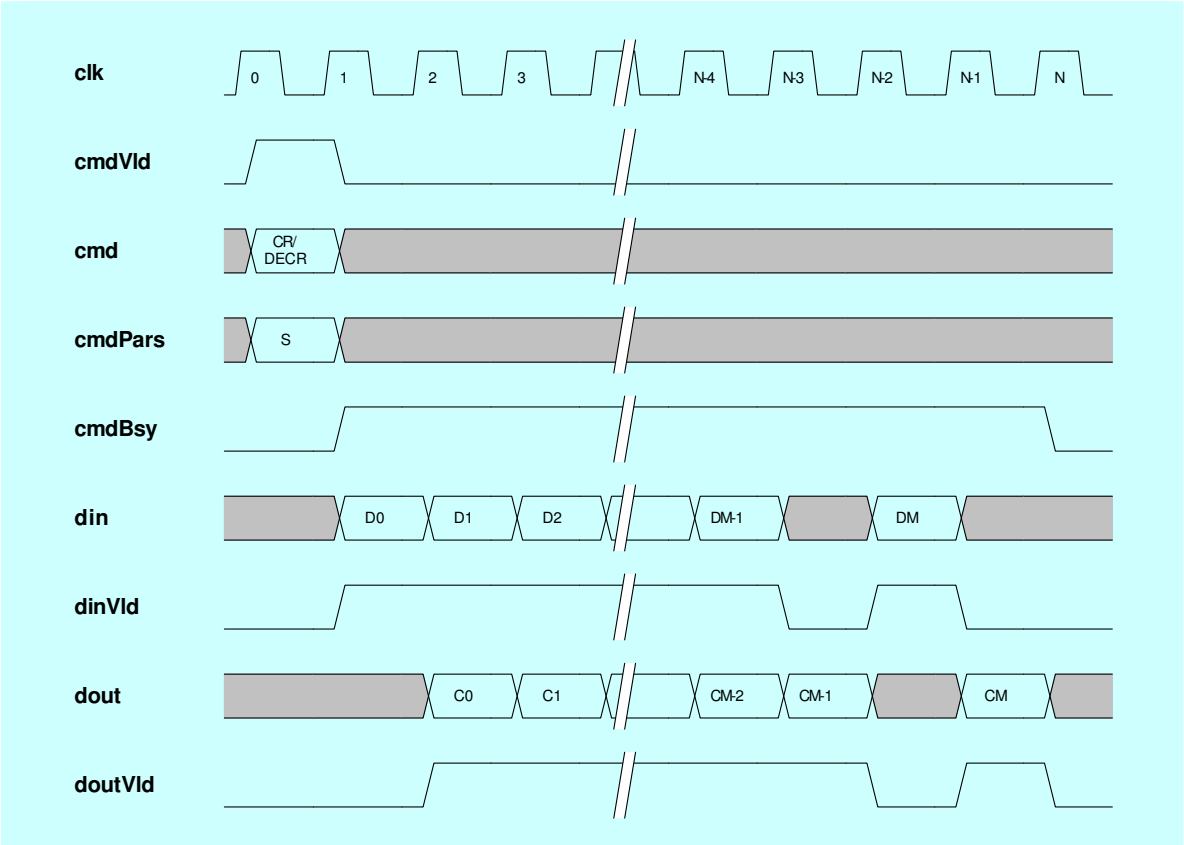

Figure 3. CRYPT/DECRYPT command Timing Diagram

For convenience, context memory can be read/written through ARC4 core. This eliminates the need for an external arbiter to memory to provide this access to memory from CPU. Figure 4 and Figure 5 depict Write and Read cycles to context memory through the command interface.

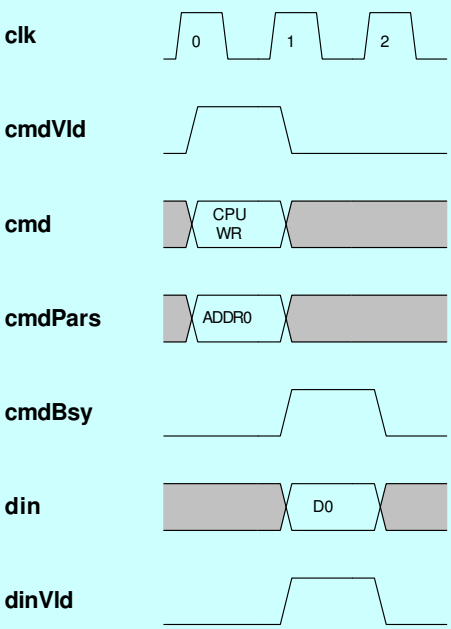

Figure 4. CPU_WR command Timing Diagram

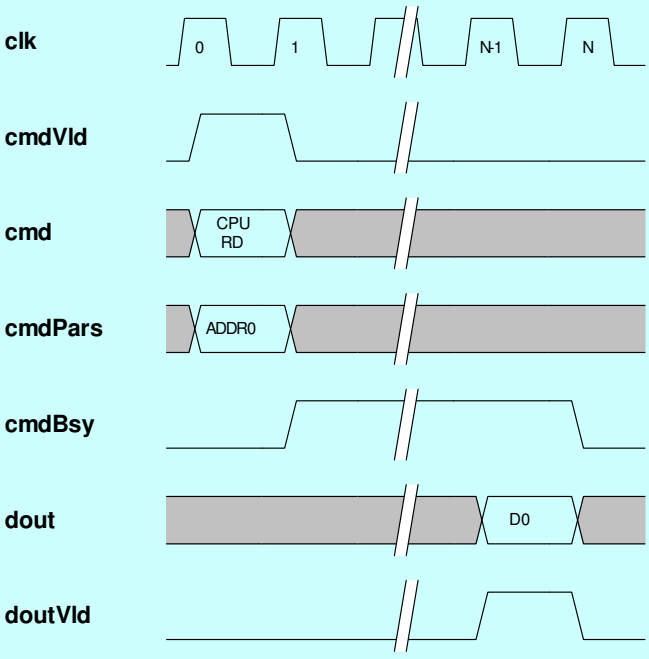

Figure 5. CPU_RD command Timing Diagram

Deliverables

- ✓ Synthesizable Verilog RTL source code
- ✓ Simulation scripts
- ✓ Self-checking Test environment
  - ➢ Test-bench
  - ➢ Test-vectors
  - ➢ Expected results
- ✓ Synthesis scripts
- ✓ User Documentation

Sales Information

For pricing information:

Esencia Technologies Inc.
2041 Mission College Blvd., Suite #100
Santa Clara CA, 95054
Tel: (408) 736-8284
Web: www.esenciatech.com
E-mail: sales@esenciatech.com

About Esencia

Esencia Technologies is a leading provider of pre-verified virtual components for consumer electronics and communication markets at competitive prices.