Scrypt Hashing ASIC

Team Members: Kyle Chynoweth, Eric Schrock

# Design Specific Success Criteria

1. Demonstrate by simulation of a Verilog test bench that the device correctly hashes cryptocurrency block headers, using a sample of blocks from the blockchains of scrypt cryptocurrency (3 points)
2. Demonstrate by simulation of a Verilog test bench that the device accepts changes to any variable (version, previous hash, merkle root, timestamp, and target) individually or together via the i2c interface (3 points)
3. Demonstrate by simulation of a Verilog test bench that scrypt core must be scalable to allow multiple cores to operate in parallel, all connected as slaves to the i2c bus (2 points)

# Verification Plan Summary

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **What to Verify** | **Design Module(s) involved** | **Verification Procedure Summary** | **DSSC Proved** | **Use in Final Demo** | **Comments** |
| **Correctly hashes PBKDF2- HMAC- SHA256** | PBKDF2\_80\_80\_128  PBKDF2\_80\_128\_32  HMAC\_SHA256\_164  HMAC\_SHA256\_212  HMAC\_SHA256\_32  SHA256 | Pre/post processing test | 1 | Only if can't show that top scrypt block operates correctly | Part of primary Scrypt hash test |
| **Correctly updates values in primary SRAM** | Top-Level  (i2c transceiver, controller, SRAM) | Input command and data over i2c, check that SRAM updates according to the command | 2 | yes | Part of primary Scrypt hash test |
| **Correctly performs primary Scrypt component hash** | SCRYPT\_BLOCKMIX  SCRYPT\_SMIX  SALSA\_20\_8 | Performs the primary Scrypt algorithm on sample | 1 | Only if can't show that top scrypt block operates correctly | Part of primary Scrypt hash test |
| **Correctly performs overall Scrypt hashing** | SCRYPT\_TOP | Input 80B block header data and compare output to known results | 1 | yes | Primary Scrypt hash test. Contains other tests. |
| **i2c correctly decodes commands** | i2c transceiver | Transmit command codes via i2c and check transceiver block outputs | 1, 2 | no | Send commands, check decoded outputs |
| **Controller correctly responds to commands** | controller | Provide decoded commands to controller, check output commands to other blocks | 1, 2 | no | Receive decoded inputs, check FSM state and outputs |
| **Correctly operates in parallel over the i2c bus** | top-level | Use the provided i2c bus to connect multiple devices in parallel. Test that they each receive commands and output hashes | 1, 3 | yes | Can re-use |

# Detailed Verification Test Breakouts

## Correct Chip Response to Encryption & Decryption Requests

* DSSC(s) Proved: 1 & 2
* Highest Level of Design Module(s) involved:
  + Total Design/Chip
* Test bench Expectations/Requirements:
  + Have temporary data registers to capture encryption results for reuse in decryption request
  + Use random number generation to get data to encrypt/decrypt
  + Emulate encryption request AHB-Lite bus transaction
  + Emulate decryption request AHB-Lite bus transaction
* No external or premade references are needed
* No pre/post processing is needed
* Main Verification Test Steps:
  1. Generate encryption request bus transaction for design
  2. Capture encrypted data result
  3. Generate decryption request bus transaction for design using prior encrypted data result
  4. Check that decrypted data result matches original raw data
  5. Repeat Steps 1-4 for multiple randomized data values

## Correctness of Blowfish Encryption (Top-level)

* DSSC(s) Proved: 3
* Highest Level of Design Module(s) involved:
  + Total Design/Chip
* Test bench Expectations/Requirements:
  + Simply use loop back test bench
* Use <http://blowfish.online-domain-tools.com/> as reference for blowfish correctness
* No pre/post processing is needed
* Main Verification Test Steps:

1. Check each/random encryption transactions from the loopback test using the above website Blowfish implementation

## Correctness of Blowfish Decryption (Top-level)

* DSSC(s) Proved: 4
* Highest Level of Design Module(s) involved:
  + Total Design/Chip
* Test bench Expectations/Requirements:
  + Simply use loop back test bench
* Use <http://blowfish.online-domain-tools.com/> as reference for blowfish correctness
* No pre/post processing is needed
* Main Verification Test Steps:

1. Check each/random decryption transactions from the loopback test using the above website Blowfish implementation

## AMBA AHB-Lite Protocol Interfacing (Top-level)

* DSSC(s) Proved: 5
* Highest Level of Design Module(s) involved:
  + Total Design/Chip
* Test bench Expectations/Requirements:
  + Have test vector of samples of each type of bus transaction and correct chip response information
  + Chip response should be verified offline by checking against standards
* No external or premade references are needed
* No pre/post processing is needed
* Main Verification Test Steps:

1. Emulate bus transaction sample from test vector
2. Check chip response against the correct response values in the test vector
3. Repeat Steps 1-2 for all entries in test vector

## Correctness of Blowfish Encryption (Encryption Block)

* DSSC(s) Proved: 3
* Highest Level of Design Module(s) involved:
  + Encryption Block
* Test bench Expectations/Requirements:
  + Random number generator for raw data generation
* Use <http://blowfish.online-domain-tools.com/> as reference for blowfish correctness
* No pre/post processing is needed
* Main Verification Test Steps:

1. Randomly generate raw data value to encrypt
2. Feed raw data into encryption block as if it came from the AMBA AHB-lite bus interface block
3. Check encrypted output data with above gold reference model
4. Repeat steps 1-3 for multiple data values

## Correctness of Blowfish Decryption (Decryption Block)

* DSSC(s) Proved: 4
* Highest Level of Design Module(s) involved:
  + Decryption Block
* Test bench Expectations/Requirements:
  + Random number generator for raw data generation
* Use <http://blowfish.online-domain-tools.com/> as reference for blowfish correctness
* No pre/post processing is needed
* Main Verification Test Steps:

1. Randomly generate raw data value to decrypt
2. Feed raw data into decryption block as if it came from the AMBA AHB-lite bus interface block
3. Check decrypted output data with above gold reference model
4. Repeat steps 1-3 for multiple data values

## AMBA AHB-Lite Protocol Interfacing (AMBA AHB-Lite Bus Interface Block)

* DSSC(s) Proved: 5
* Highest Level of Design Module(s) involved:
  + AMBA AHB-Lite Bus Interface Block
* Test bench Expectations/Requirements:
  + Have test vector of samples of each type of bus transaction and correct response information
  + Use different dummy data for transaction payloads to check that intended values are actually getting received/sent
  + Response information should be verified offline by checking against standards
* No external or premade references are needed
* No pre/post processing is needed
* Main Verification Test Steps:

1. Emulate bus transaction sample from test vector
2. Check block response against the correct response values in the test vector
3. Repeat Steps 1-2 for all entries in test vector