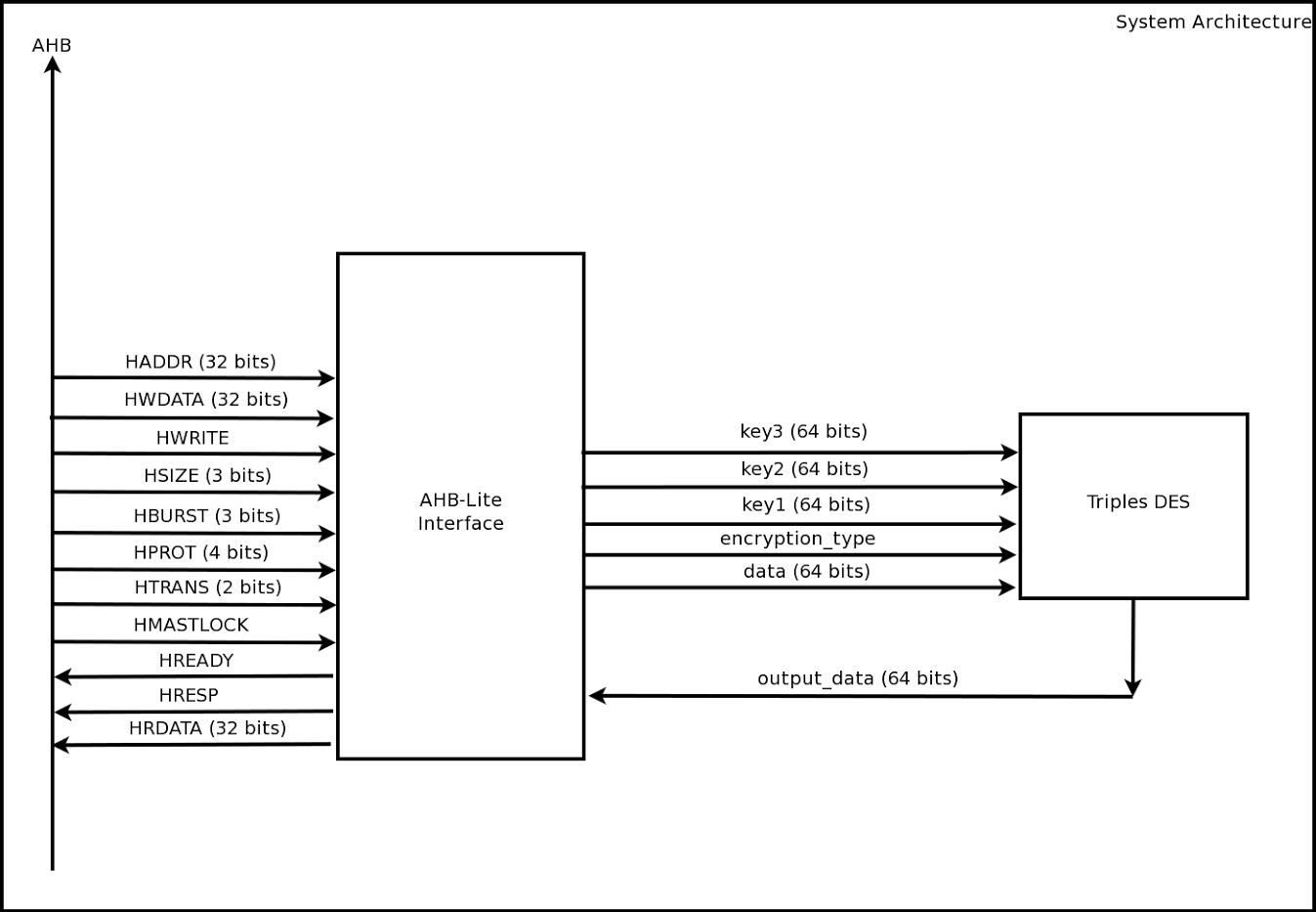
Triple DES Encryption / Decryption for USB 2.0 Specification Verification Plan

Team Members: Seth Bontrager, Anthony Kang, Eric Murphy, Isaac Sheeley

**Architecture**



**AHB-Lite Interface:** Top level block that deals with sending and receiving of data from the bus. This module will contain the IO controllers from our design (Controller 1 and Controller 2). Data is sent in 32-bit chunks, with the keys, data, and encryption type being sent in the initial transaction.

**Triple DES:** Top level block that deals with the encryption and decryption of our data. This block will take care of the key generation (Keys will be input in the beginning of a transaction), DES blocks (Three total for Triple DES), and pipelining.

**Fixed Success Criteria**

1. Test benches exist for all top level components and the entire design. The test benches for the entire design can be demonstrated or documented to cover all of the functional requirements given in the design specific success criteria. (2 pts)
2. Entire design synthesizes completely, without any inferred latches, timing arcs, and, sensitivity list warnings (4 pts)
3. Source and mapped version of the complete design behave the same for all test cases. The mapped version simulates without timing errors except at time zero (2 pts)
4. A complete IC layout is produced that passes all geometry and connectivity checks (2 pts)
5. The entire design complies with targets for area, pin count, throughput (if applicable), and clock rate. The final targets for these parameters will be determined by course staff based on your design review. Failure to reach any of the targets will result a score of 1 out of 2 provided that you are within 50% on area, 10% on pin count, and 25% on throughput. Doing worse in any category will result in a score of 0. (2 pts)
   1. Area: 4.0mm x 4.0mm
   2. Pin Count: 115 (112 for the System Architecture + power, ground, clock)
   3. Clock Frequency: 200 MHz

**Design Specific Success Criteria**

1. Demonstrate by utilizing a known, working python script that the output of the design both encrypts and decrypts according to the 3DES algorithm (1 pt)
2. Demonstrate by simulation of verilog test benches that the complete design is able to utilize pipelining (1 pt)
3. Demonstrate by simulation of verilog test benches that the complete design is able to successfully implement 3DES encryption (2 pts)
4. Demonstrate by simulation of verilog test benches that the complete design is able to successfully implement 3DES decryption (2 pts)
5. Demonstrate by simulation of verilog test benches that the complete design is able to implement an IO controller for the AHB-lite (2 pts)

**Verification Plan Summary**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **What to Verify** | **Design Module(s) involved** | **Verification Procedure Summary** | **DSSC(s) Proved** | **Use in Final Demo** | **Comments** |
| **Correct chip response to encryption & decryption requests** | Top Level | Use result from encrypted output as input to decryption via test bench | DSSC 3, 4, 5 | Yes | Use temporary resisters in test bench that save results from encryption to use in decryption |
| **Correctness of 3DES Encryption** | Top Level | Compare results to working python code | DSSCs 1, 2, 3 | Yes | Use temporary registers in test bench to grab encrypted output |
| **Correctness of 3DES Decryption** | Top Level | Compare results to working python code | DSSCs 1, 2, 4 | Yes | Use temporary registers in test bench to grab decrypted output |
| **AMBA AHB-Lite Protocol Interfacing** | Top Level | Test bench provides different bus transactions | DSSC 5 | Yes | Should be able to reuse test bench from I/O controller 1 and 2 |
| **Correctness of 3DES Encryption** | DES blocks with a input of encryption\_type = 1 | Compare results to working python code | DSSCs 1, 2, 3 | Only if can’t show using top level | encryption\_type = 1 corresponds to encrypting |
| **Correctness of 3DES Decryption** | DES blocks with a input of encryption\_type = 0 | Compare results to working python code | DSSCs 1, 2, 4 | Only if can’t show using top level | encryption\_type = 0 corresponds to decrypting |
| **AMBA AHB-Lite Protocol Interfacing** | IO Controller 1 and 2 | Test bench provides different bus transactions | DSSC 5 | Only if can’t show using top level | Should be able to reuse test bench from I/O controller 1 and 2 |

**Detailed Verification Test Breakouts**

**Demo Tests**

**Correct Chip Response to Encryption & Decryption Requests**

* Shown in Demo: Yes
* DSSC(s) Proved: 3, 4, 5
* 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 (i.e. Have Professor Johnson choose a random number) for encryption / decryption
  + Emulate encryption request AHB-Lite bus transaction
  + Emulate decryption request AHB-Lite bus transaction
  + Capture the generated keys and compare them to known output from python script
* No pre/post processing is needed
* Use python script to verify results
* Main Verification Test Steps:

1. Verify that when encryption\_type = 1, the encryption process begins
2. Verify that when encryption\_type = 0, the decryption process begin
3. Verify that when encryption\_type = 1, the correct keyset is generated
4. Verify that when encryption\_type = 0, the correct keyset is generated

**Correctness of 3DES Encryption (Top-level)**

* Shown in Demo: Yes
* DSSC(s) Proved: 1, 2, 3
* Highest Level of Design Module(s) involved:
  + Total Design/Chip
* Test bench Expectations/Requirements:
  + Have several test cases that encrypt a variable number of data blocks
  + Encrypt with several different keys in different test cases
  + Output should be verified using known working python script
* Use handmade Python script as a reference for 3DES correctness
* No pre/post processing is needed
* Main Verification Test Steps:

1. Use testbench to capture encrypted output
2. Compare encryption output with output from known correct software implementation

**Correctness of 3DES Decryption (Top-level)**

* Shown in Demo: Yes
* DSSC(s) Proved: 1, 2, 4
* Highest Level of Design Module(s) involved:
  + Total Design/Chip
* Test bench Expectations/Requirements:
  + Have several test cases that decrypt a variable number of data blocks
  + Decrypt with several different keys in different test cases
  + Output should be verified using known working python script
* Use handmade Python script as a reference for 3DES correctness
* Main Verification Test Steps:

1. Use testbench to capture decryption output
2. Compare decryption output with output from known correct software implementation

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

* Shown in Demo: Yes
* 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 will checked against standards
* No pre/post processing is needed
* Main Verification Test Steps:

1. Simulate bus transaction sample
2. Check chip response against correct response values
3. Repeat steps 1 and 2 for all the different transaction types

**Backup and Sub-Module Tests**

**Correctness of 3DES Encryption (Encryption Block)**

* Show in Demo: Only if can’t show using top level
* DSSC(s) Proved: 1, 2 ,3
* Highest Level of Design Module(s) involved:
  + DES blocks with a input of encryption\_type = 1
* Test bench expectations/requirements:
  + Use temporary registers in test bench to grab decrypted output
* No pre/post processing is needed
* Use python script as reference for 3DES correctness
* Main Verification Tests Steps:

1. Use testbench to capture output of encryption
2. Check to confirm that the output of the encryption blocks is the same as the output that is provided from the python script

**Correctness of 3DES Decryption (Decryption Block)**

* Show in Demo: Only if can’t show using top level
* DSSC(s) Proved: 1, 2, 4
* Highest Level of Design Module(s) involved:
  + DES blocks with a input of encryption\_type = 1
* Test bench expectations/requirements:
* No pre/post processing is needed
* Use python script as reference for 3DES correctness
* Main Verification Tests Steps:

1. Use test bench to capture output of decryption
2. Check to confirm that the output of the encryption blocks is the same as the output that is provided from the python script

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

* Shown in Demo: Yes
* 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 will checked against standards
* No pre/post processing is needed
* Main Verification Test Steps:

1. Simulate bus transaction sample
2. Check chip response against correct response values
3. Repeat steps 1 and 2 for all the different transaction types