# Full Project Proposal

ECE 337

Seth Bontrager
Anthony Kang
Eric Murphy
Isaac Sheeley

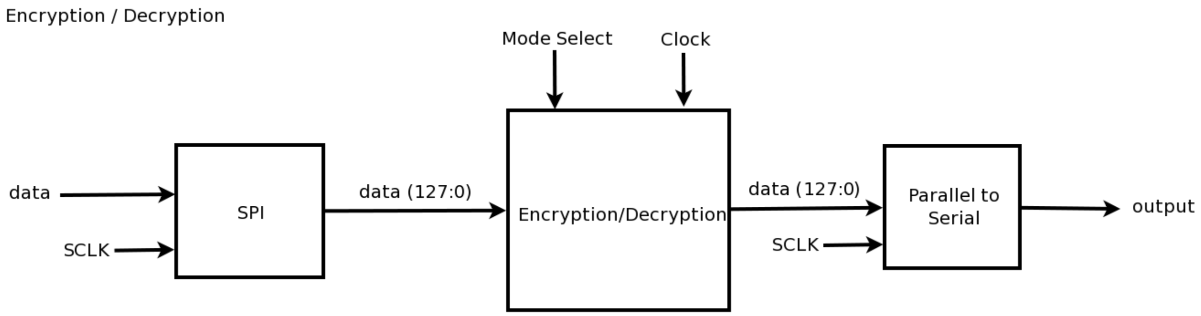
Thursday (11:30 - 2:20)
Due: 3/13/15
TA: Utpal Mahanta

# Executive Summary

Our team wants to design and build a hardware implementation of the 3DES algorithm. The sending and receiving of encrypted data is an integral part of nearly all forms of commerce and internet communications. Our design of 3DES is important because many financial institutions still use the encryption standard, therefore developing a more efficient or faster implementation could be very beneficial. Our algorithm will be unique in the sense that we will be optimizing for speed of encryption and decryption. This will be accomplished by optimizing certain functional blocks at a gate level. The functional blocks of choice will be later determined by the estimated benefit of implementing them at a gate level. The 3DES algorithm is appropriate for ASIC because it's creators developed it will hardware implementation in mind. Because of this fact, 3DES can successfully be implemented in a hardware design to allow for the encryption and decryption of data without having to consume resources that a software based implementation would otherwise devour. In order to implement this design, we will need knowledge of the DES algorithm, a development and testing environment, work hours for developing and testing the design, and support from the course staff as we may encounter difficulties that are currently unforeseen. The remainder of this proposal goes over the details of implementing 3DES, such as the design specifications, a system usage diagram that shows a very high-level block diagram of the overall algorithm along with a diagram for one round of processing. It also details the operation characteristics that describes the functions that will be performed by our chip and identifies the general type of commercial part to which our final chip would be connected.

# Design Specification

## System Usage Diagram



High Level Block:

This block encompasses the inputs, outputs, and the functionality of our design.

Port Descriptions:

| Signal | Direction | Description |
|---|---|---|
| SCLK | input | The SPI clock, used to clock in values |
| data | input | Data that is being clocked into the system (1 bit at a time) |
| Mode_Select | input | Selects the mode for the system to use (Encrypt or Decrypt) |
| Clock | input | The system clock, used to clock the overall design |
| output | output | Data is that being output (1 bit at a time) |

# Operational Characteristics

Encryption Block

This is the block that implements each round of the algorithm.

Port Descriptions:

| Signal | Direction | Description |
|---|---|---|
| clk | input | system clock signal with a 50% duty cycle |
| n_rst | input | Asynchronous active low system reset signal. |
| plaintext[127:0] | input/output | plaintext message for encrypted, decrypted message from decryption |
| ciphertext[127:0] | input/output | encrypted text input to be decrypted, encrypted text output from encryption |
| keys[0:3][0:55] | input | 3 keys, one for each round of DES |

DES Block

This block implements the 3 rounds of DES in triple DES

| Signal | Direction | Description |
|---|---|---|
| plaintext[127:0] | input/output | left 32 bits of input block |
| ciphertext[127:0] | input/output | right 32 bits of input block |
| round_key[0:55] | input | round key used for encryption/decryption |

Round Block

This is the block that houses the functional blocks for each step of the algorithm.

Port Descriptions:

| Signal | Direction | Description |
|---|---|---|
| left[31:0] | input | left 32 bits of input block |
| right[31:0] | input | right 32 bits of input block |
| round_key[0:48] | input | round key used for encryption/decryption |

Expand Block

Expansion block that uses a expansion permutation to expand 32 bits in to 48 bits.

| Signal | Direction | Description |
|---|---|---|
| right[31:0] | input | right 32 bits of input block |
| data[47:0] | output | expanded 48 bits of input block |

Substitution Block

Substitution block uses the expanded input block to substitute values from a lookup table.

| Signal | Direction | Description |
|---|---|---|
| data[47:0] | input | expanded 48 bits |
| data_out[31:0] | output | substituted 32 bits |

Permutation Block

Permutation block simply permutes the 32 bits using a preset permutation box

| Signal | Direction | Description |
|---|---|---|
| data[31:0] | input | substituted 32 bits |
| data_out[31:0] | output | permutated 32 bits |

KeyGen Block

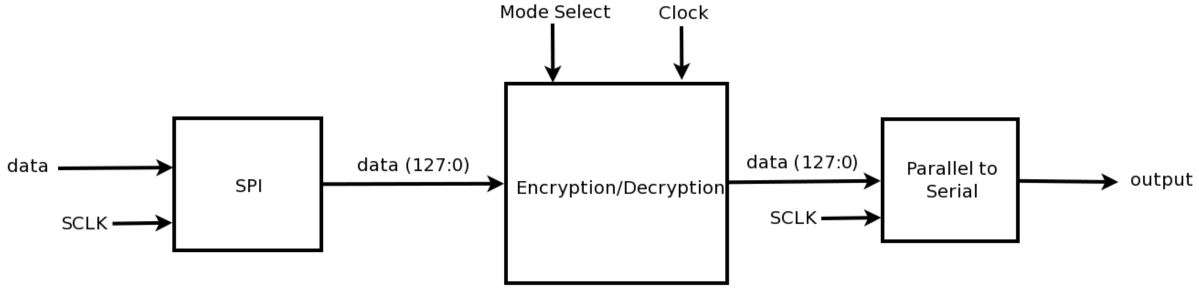
Key Generation block that generates the round key but doing a 56 bit permutation and a 48 bit contraction

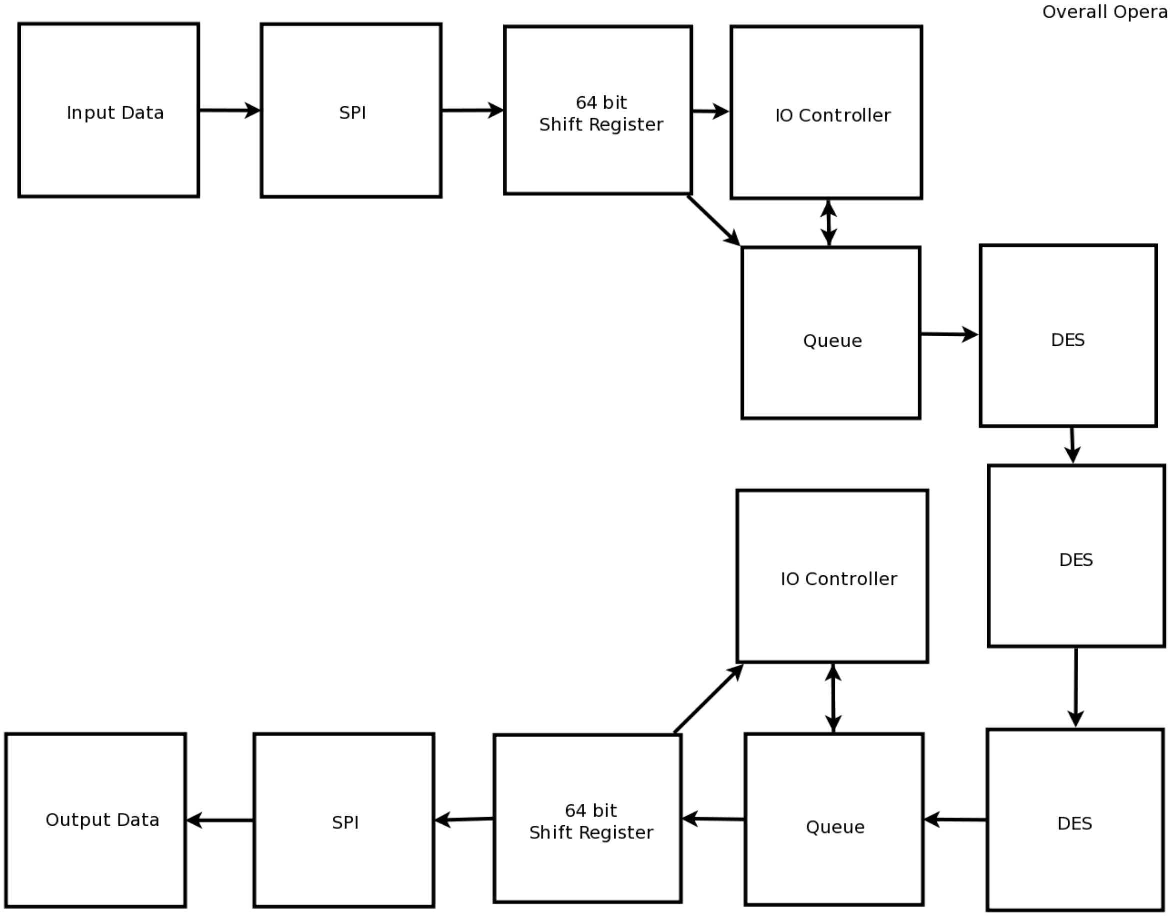
| Signal | Direction | Description |
|---|---|---|
| user_key[56:0] | input | user supplied 56 bit key |
| gen_enable | input | signals block to shift in user key |
| gen_next | input | signals the block to generate the next key |
| prev_key[56:0] | input | previous key, as next key depends on previous key |
| round_key[48:0] | output | 48 bit permuted/contracted key |

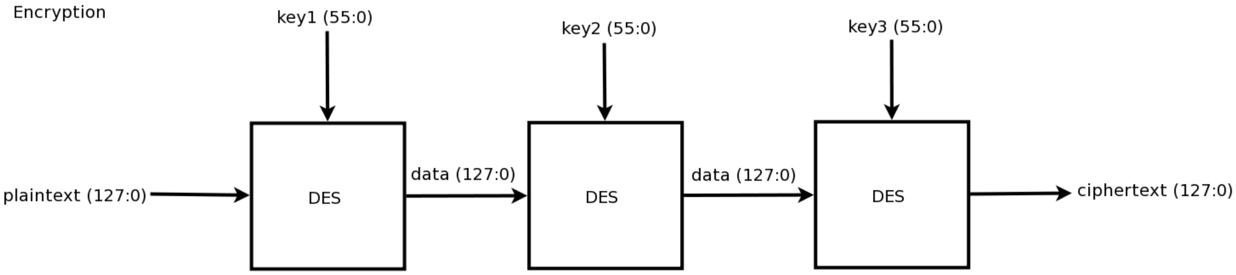
# Requirements

The use of Triple DES in encrypted network communication requires high speed to minimize the time it takes to encrypt the information and then decrypt it at the other end. The speed at which information can be sent is very critical in many industries today, and the methods used to encrypt the information must be fast in order to not create a bottleneck in the transmission of information. The primary optimization objective of our project will be to maximize speed. We will optimize the design for speed by implementing certain functional blocks at a gate level and pipelining several steps in the algorithm so the next block is being encrypted directly behind the previous one. Optimizing our design for area will be a secondary objective. Our target for the area will be the initial 1.5 mm x 1.5 mm for now until we get a better idea of the area required for our design. The pin count for our design will be 6 pins. Since data is going to be input and output in a serial manner, fewer pins are needed than in other kinds of designs. We will utilize 2 pins for the SCLK (two separate clocks are used to reduce any potential problems). Two pins will be used for data (data coming in and data coming out). A pin is used to select the mode of operation (encryption or decryption), and a final pin is used for the overall system clock (since we will want a faster clock for the overall design).

# Design Architecture


Encryption / Decryption

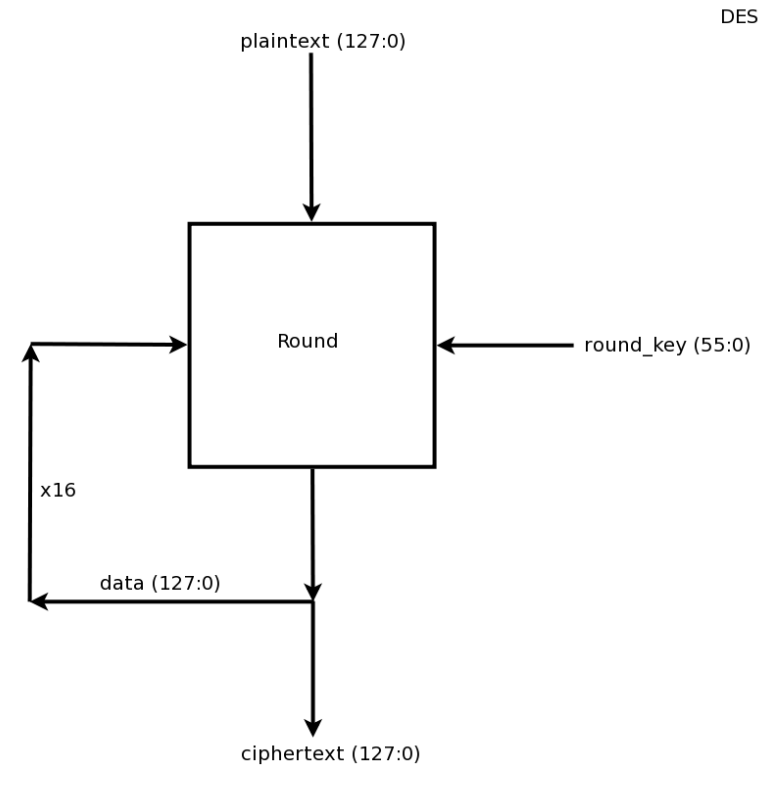

Overall Operation


Encryption

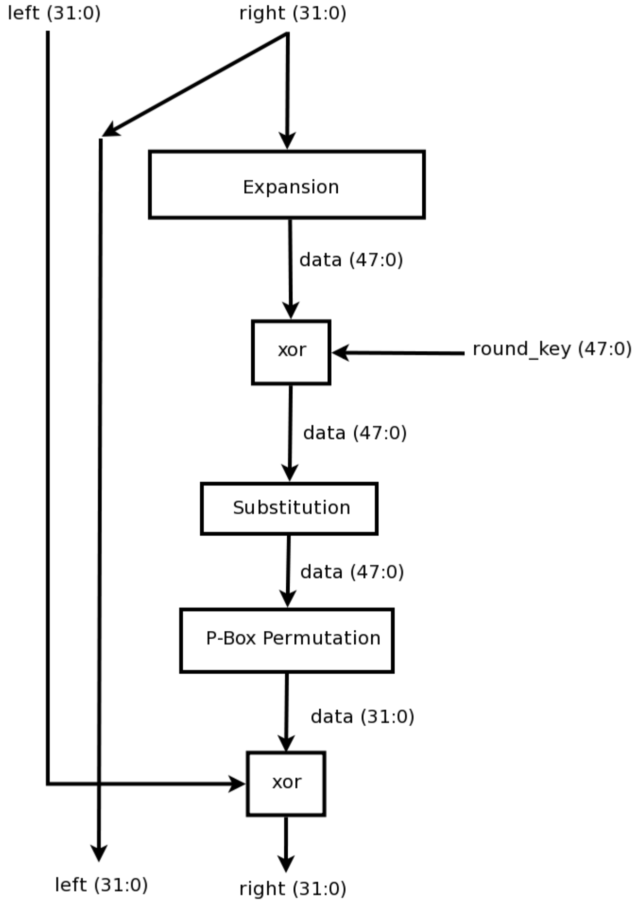
Pseudo-code for general 3DES algorithm:

```
while data available

   Load 64 bit block

   for DES in 3DES
          for round in range(16)
                   split block into 32bit halves
                   expand right half to 48 bits
                   xor right with round key
                   run right through s-box substitution
                   run right through p-box permutation
                   xor right with left half
                   swap left and right
                   combine into 64 bit block

          swap left and right

   output encrypted 64 bit block
```

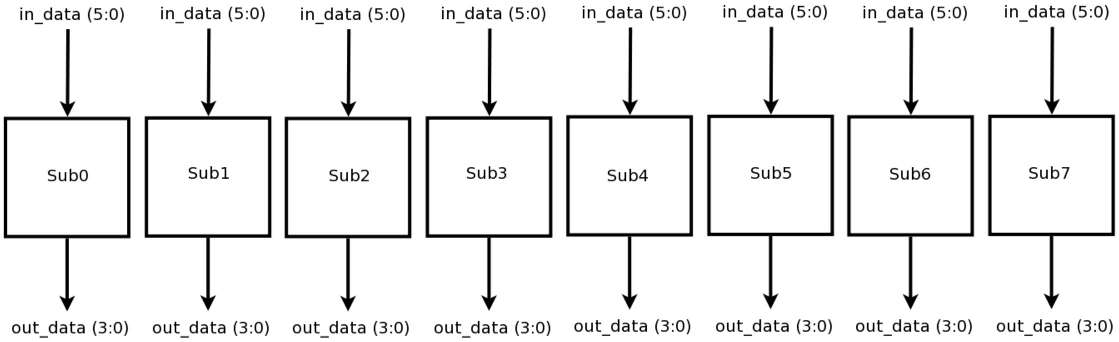
**NOTE
Decryption follows same algorithm



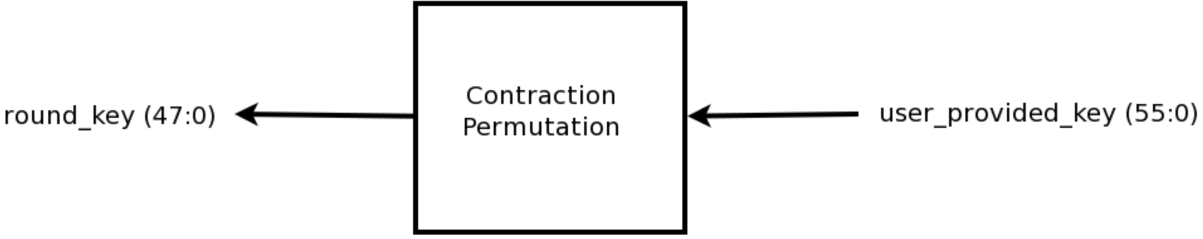
One Round of DES



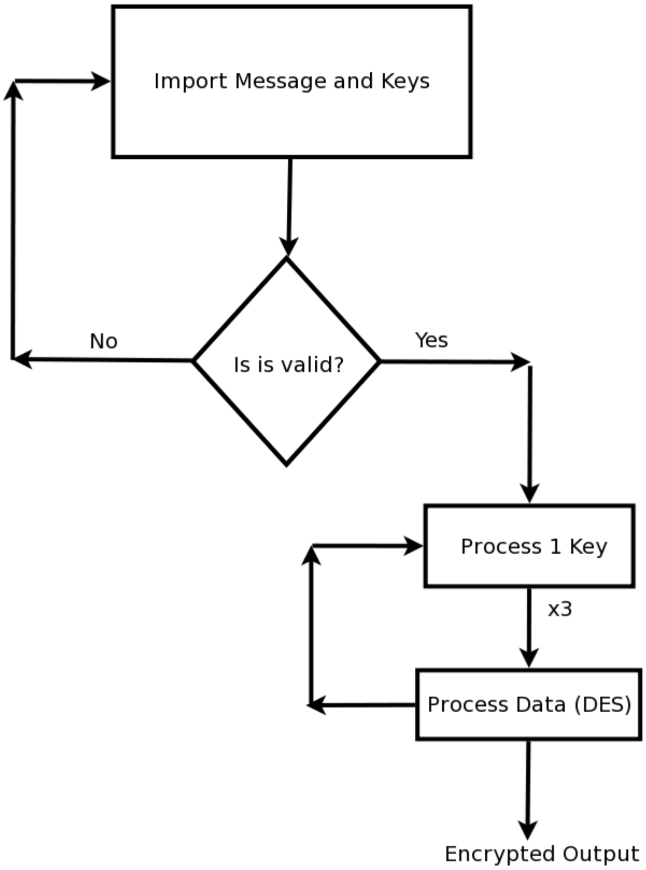
Substitution Step



Key Contraction



Flowchart of Chip Operation



# Projected Timeline

| Overall Timeline | |
|---|---|
| March 23 - March 29 | Programming simpler portions of the projects (Including test benches) |
| March 30 - April 5 | Start programming the more difficult blocks of the project. Start working on the SPI interface. |
| April 6 - April 12 | Begin interfacing separate blocks together (Writing test benches to verify connectivity). Look in to timing analysis to reach desired speeds. |
| April 13 - April 19 | Interface with input and output. Start implementing the final connections. Testing design's input and output along the way. |
| April 20 - April 26 | Final testing and further optimizations to the design. |
| April 27 - May 1 | Prepare for final presentation and report. |

| Individual Contributions | |
|---|---|
| Seth Bontrager | Generation of round keys (Gate level design), Test bench for round key generation, Input and Output (Accepting input via SPI), Test bench to verify Input and Output, Overall test bench design |
| Anthony Kang | P-box permutations, Test bench to verify the p-box permutation, DES round wrapper file, Test bench to verify working DES round |
| Eric Murphy | S-box substitutions, Test bench to verify substitutions, SPI, Test bench to verify working input and output for SPI, Task Monitor and Task negotiator |
| Isaac Sheeley | Key expansion algorithm, Test bench to verify correct expansion of keys, DES wrapper file, Test bench to verify working DES encryption and decryption, Overall test bench design |

# Success Criteria

- Fully working test benches for for all top level components and the entire design
- Entire design synthesizes completely with no latches, timing arcs, or sensitivity list warnings
- Source and mapped version of the design behave as expected
- A complete IC layout is produced and all geometry and connectivity checks
- 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 in 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 out of 2

- Demonstrate by simulation of verilog test benches that the complete design is able to successfully implement 3DES encryption

- Demonstrate by simulation of verilog test benches that the complete design is able to successfully implement 3DES decryption

- Demonstrate by utilizing a known, working python script that the output of the design both encrypts and decrypts according to the 3DES algorithm

- Demonstrate by simulation of verilog test benches that the complete design is able to utilize pipelining to increase speed

- Demonstrate by simulation of verilog test benches that the complete design is able to implement an IO controller for the SPI