DES
(Data Encryption Standard)

DES was originally developed by IBM in 1977 to encode binary data. It uses 56 bit keys to cypher blocks of 64 bits of raw data into 64 bits of encrypted data to be transmitted.

The DES algorithm is split up into 19 distinct stages. The first and the last stages consist of a simple transposition of bits, independent of the key used. The second last stage consist of exchanging the first 32 bits fifth the last 32 bits. The remaining 16 stages perform bit transfers between the 64 bits of the data based on certain parameters of the encryption key. The decryption is done by performing the opposite steps in the reversed order to generate the original text.

We propose to build in VHDL a decrypting unit capable of deciphering 56 bit DES encoded data in a reasonable amount of time. This will be done by initially develop in software the DEA algorithm. Afterwards, use VHDL to design both an encoding and a decoding unit, which will be used as a testbench for the decrypting unit. Then, using brute force attack, use the decryption unit to decipher ASCII data.

DES was chosen as a target for decyphering for the following reasons:

- DES encryption is mostly based on bit manipulation rather than arithmetic operations (like RSA, WPA, etc..), therefore the use of hardware Vs software to perform a brute force attack should generate a significant speed up
- The implementation on FPGA has already been made, therefore it is a feasible project
- 56 bit DES encryption is obsolete and no longer used as an encryption standard (It has been replaced by Triple DES, roughly the same algorithm but using 192 bit keys), therefore developing tools to decrypt this algorithm dos not compromise any confidential data and is not unethical

Components to be built

Auxiliary:

- Software simulations
- An encryption unit
- A plain ASCII message to encrypt

Primary:

1- A biased key generator (see below)
2- A decrypting unit
3- A DATA coherence checking unit which will compare the decyphered data to actual data to judge its coherence. If the data encrypted was ASCII text, it will compare the 7 decrypted byted to ASCII text characters and reject incoherent. It couls also compare the data with known file or packet headers (to be decided)

Timing estimations:

A brute force attack on 56 bit encrypted data should take 2^56 = 72057594037927936 trials at most, 2^55 = 36028797018963968 on average. Since the 56 bit key is actually a human generated sequence of ASCII characters, the task is reduced to 128^7 = 562949953421312. Based on basic psychology, we can anticipate that it will mostly contain regular characters and numbers (who really used '#~/>!' in a password...?) , therefore reducing the primary test pool to (26 regular letters + 26 capital letters + 10 numbers + space) = 65^7 = 4902227890625. Since the encryption is done linearly, we should be able to pipeline the deciphering therefore testing one case every clock pulses. On a 50 MHz clock, this would take only 98 seconds best case, 1 441 151 seconds (16.7 days) absolute worst case. Since the simulations will be running on a PC clocked in the GHz, the real time consumed might be even further reduced.

Limitations

Brute Force decryption is great because it guarantees results if all the test cases are exhausted, no matter how long the encryption key. The longer the encryption key and the more irregular the encrypted data, the more time/hardware demanding the decryption process will be.

Therefore, we propose to limit ourselves to encryption keys made out of common 'human' characters (see above). Furthermore, AI systems might be required to match test every deciphered test key cases with coherent data, which is beyond the scope of this course.
We will restrict ourselves to assuming that the block of encrypted data presented is either plain ASCII text or known packet headers. Validating the generated results will therefore be greatly simplified to matching the potential data with an ASCII LUT or with the (known) source and destination IP fields (last 64 bits of a TCP/IP packet header) of a "captured over DES encrypted WiFi" packed (in reality generated by our encrypter.