Digital Logic Design

Final Project Spring 2022

Design of a DES Cracker Project Amendment

The goal of the project is to get you to see a complete system and implement it. It is also important to utilize all the skills we utilized throughout the semester. However, as explained in class, engineering is about weighting problems and making sure there are more advantages than disadvantages. This project is no different in that the key is generated from the Up/Down/Load (UDL) counter without parity. This may interfere with the process of "crack" the solution appropriately.

Like any project in industry, White Papers are typically designed to help alleviate concerns. This is a solution or White Paper Ross Thompson created by making a new module that fixes the counter out of the UDL counter. There are many solutions to these problems, but we are illustrating a quick fix that will help many. You are welcome to try your own fix and, again, there are many solutions.

Parity Generation

In this SystemVerilog Hardware Description Language (HDL), a cool *for-generate* statement is utilized to generate the parity every 8 bits as dictated by the DES FIPS 46-3 standard. The *for-generate* statement looks at every 7-bits and generates the appropriate parity on every 8th bit.

```
module genParity(input logic [6:0] in, output logic [7:0] out);
  assign out[0] = ~^in;
  assign out[7:1] = in;
endmodule

module genParity8(input logic [55:0] in, output logic [63:0] out);
  genvar                                    index;
  for(index = 0; index < 8; index++) begin
      genParity genParity(.in(in[7*index +: 7]), .out(out[8*index +: 8]));
  end
endmodule
```

Implementation

In order to implement the design, you should utilize this after the UDL counter and then before your DES block. It should look like the code shown below where the output of the counter (i.e., count), goes into the genParity8 block and outputs the key correctly. Another nice thing that was done here is that the UDL counter now is 56 bits and that will save you on your FPGA area. I hope you see this is an easy and effective solution and helps alleviate problems with some cracking.

You can just utilize the code above and insert it into your top design and it should work provided you instantiate it appropriately as shown below. The output `DESkey_input` should go into your DES input for the key.

Although we try everything in our power to create a project that does not have problems, it sometimes exhibits problems. However, with good engineering and using our brain, we can solve problems more efficiently and effectively. Please feel free to contact us if you have questions about this solution. Again, please feel free to use your own solution, if needed.

```
UDL_Count #(56) p0 (clk, 1'b0, UP, down, reset, in, count);
// convert key by adding parity.
genParity8 genParity8(.in(count), .out(DESkey_input));
```