Project for Hardware-Software Codesign

Assigned: Tue., Nov 26, 2105
Due: Dec. 2nd, 2015

Description: Write a C program to retrieve 2048 timing values and generate a bitstring

IMPORTANT NOTE: My EvalFUOutputs.vhd file had a small bug that is now fixed. If you do not have your module running, you should use mine to build the programming bitstream.

- Use the following 2-vector sequences to generate the 2048 timing values:
- The 1st vector of each pair is always 16 zeros, e.g., "0000000000000000"
- The 2nd vector of each pair counts from 0 to 255 in the left-most 8 bits and counts from 255 downto 0 in the right-most 8 bits. For example the 2nd vector of the first vector pair is "0000000011111111". The 2nd vector of the second vector pair is "0000000111111110". The 2nd vector of the third vector pair is "0000001011111101". And so on. Note that the right half of the 2nd vector is always the complement of the left half.
- Each 2-vector sequence produces exactly 8 timing values (from the 16 outputs of the functional unit). Therefore, 256 2-vector sequences produce exactly 2048 timing values.

Store the 2048 values in an array in your C program. Once all of them are generated, apply a modulus of 32 to the values. For example, a timing value of (248 mod 32) = 24 since 248 - 32*7 = 248 - 224 = 24.

Enrollment:

- Compute a bitstring by pairing up neighboring timing values and generating a '1' in the bitstring if the first value is larger than the second and a '0' otherwise.
- Skip pairings when the difference of the values is less than a threshold of 5.
- Record a helper data string that has a '1' when the absolute value of the difference is larger than the threshold of 5 and a '0' otherwise.
- For example, if your initial sequence of values after applying the modulus is 5 10 31 24 15 16, you would:
  Generate a '0' when you compare 5 with 10, and store a '1' in the helper data string,
  Generate a '1' when you compare 31 with 24 and store a '1' in the helper data string,
  Skip the next pairing since the difference 15-16 = -1, and abs(-1) is NOT >= 5. Store a '0' in the helper data string.

Regeneration:

- Compute a bitstring again this time using the helper data from enrollment, i.e., generate a bit for ONLY those timing value pairs where the helper data bit is '1'. You should be able to reproduce the exact same bitstring. If not, increase the threshold until you can.

Report Requirements:

1) Demo this in class by printing out the bitstrings generated during enrollment and multiple regenerations. How many bits were you able to generate?