ECE459/559 Secure & Trustworthy Computer Hardware Design, Spring 2017
Lab 1

For all labs:

- Use a Digilent Nexys-4 board with Xilinx Vivado 2014.4
  NOTE: Vivado 2014.4 WebPACK can be installed for free on your own laptop or PC. Also, Vivado 2014.4 is installed on the PCs in the Digital Systems Laboratory (MK224).

- Follow the general design and implementation process using Xilinx Vivado WebPACK:
  1. *Capture* the design using VHDL code.
  2. *Simulation* is run using a VHDL test bench to analyze, verify and debug the functionality of VHDL design code.
  3. *Synthesis* maps the design onto the CLBs and other functional blocks that exist on the Artix-7 FPGA (the "FPGA architecture primitives") .
  4. *Constraints Wizard* (post-Synthesis) provides an interface to update the XDC constraints for mapping FPGA I/O pins to top-level ports in your design. These constraints can also be updated by manually editing the XDC constraints file (*.xdc).
  5. *Edit Timing Constraints* (post-Synthesis) provides an interface to update the XDC timing constraints, including constraining the clock period. These constraints can also be updated by manually editing the XDC constraints file (*.xdc).
  6. *Implementation* (place & route) figures out where on the FPGA each primitive block goes and how to connect them. Specifically, the CLBs to be used, what logic each must implement, and how these CLBs must be connected on the Artix-7 FPGA.
  7. *Generate Bitstream* converts the implemented design to a programming file (.bit) to be uploaded to the BASYS3 board.

- Write a short (2 page minimum) report detailing your result and **include signature from TA or professor showing successful demonstration of a working design**. Signature page can be separate individual page (usually part of a title page) of lab report allowing more space for the remainder of the write-up.

***Task 0*:** Study the associated papers describing the SIMON encryption/decryption engine to gain a better understanding of 1) single round encryption/decryption logic and 2) the key generation algorithm. For this lab exercise, you will be implementing these two portions of a SIMON engine (the rest provided). Note that the SIMON encryption/decryption engine you are completing in this lab works with 32-bit block size and 64-bit key size.

Also, take time to familiarize yourself with the VHDL design files provided in the sources folder of the file Lab1_files.zip. The "top" file for your completed project is simon_top.vhd which has been setup to interface to a Windows PC via the UART to USB bridge on your Nexys-4 board. You will be completing two of the VHDL design files included in the source folder: ROUND_CIPHER.vhd (implements one round of the SIMON cipher) and KEY_EXPANSION.vhd (implements the SIMON key expansion algorithm). All other files can be left as they are. Further, you can open the provided project file (lab1.xpr) using Vivado 2014.4.

***Task 1*:** Update the files ROUND_CIPHER.vhd and KEY_EXPANSION.vhd to complete your SIMON encryption/decryption engine. Details for the SIMON algorithm are provided in the two papers included with the Lab1 documents on Blackboard – use these papers as a guide for your work.

*** Deliverable: Once completed (and verified via following tasks), include your updated VHDL source code for both ROUND_CIPHER.vhd and KEY_EXPANSION.vhd as an appendix in your lab report.**

***Task 2*:** Some test bench files are available as part of the Vivado project in Lab1_files.zip:

uart_top_test.vhd – For testing the provided UART controller. Likely won't need this.
simon_top_tb.vhd – For testing entire design, including UART. Likely won't need this.
encrypt_tb.vhd – For testing the provided encryption engine. **SHOULD USE THIS!**

Explore the test benches as they are useful learning tools and will be helpful for simulating and verifying your completed design. For more information on simulating VHDL designs using Vivado, please refer to the Vivado tutorial posted on Blackboard.

Using encrypt_tb.vhd, simulate your encryption engine showing that it can successfully encrypt 32-bit blocks of data according to the SIMON algorithm. This is where you verify your updated design files, ROUND_CIPHER.vhd and KEY_EXPANSION.vhd.

*** Deliverable: Take a snapshot of your simulated result, save it as an image and include in your lab report.**

***Task 3*:** Following steps outlined in the Vivado tutorial, synthesize, implement and generate a bit file for testing your completed SIMON design on the Nexys-4 FPGA board.

Important considerations:

a) Switches SW0 and SW1 are setup on your Nexys-4 FPGA board to be used to control

your SIMON encryption/decryption engine (refer to Nexys4_Master.xdc).

SW0: cipher_en – When in the "up" position (or ON), SIMON is enabled for either encryption or decryption. When in the "down" position it simply passes data through the FPGA, skipping the encryption/decryption logic altogether. (Pass through is useful for testing the communication logic)

SW1: enc_decN – When in "up" position the engine is set for encryption and is set for decryption otherwise. Note that SW1 is essentially ignored if SW0 is in the "down" position, indicating pass through.

b) LEDs TX and RX (near where your plug in the USB cable) should glow yellow when passing data through the SIMON engine. Good to check for "first-order" troubleshooting.

c) You are provided with a small command line program (fpgacm.exe) for sending small PGM image files from a Windows PC to your FPGA. If you are using Linux, talk to the instructor!

Usage of the command line program:

fpgacm port_num <input_file> <output_file>

where port_num is the number of the COM port (via USB) your FPGA connects to on the PC, <input_file> is the name of the PGM image file to be sent, and <output_file> is the file received from the FPGA once the SIMON engine has completed.

To bring up the command prompt in Windows, type "cmd" in the "Search programs and files" field at the bottom of the Start menu and hit ENTER. You will need to cd to the working Lab1 directory you are using for this exercise.

***Task 4*:** Program your FPGA (follow steps in Vivado tutorial) to implement your complete SIMON design. Once programmed, encrypt the two PGM image files provided Peyton.pgm and PowerT.pgm and save them as Peyton_USERenc.pgm and PowerT_USERenc.pgm, respectively. USER should be replaced by your UTK netid.

NOTE: You should also use the FPGA to decrypt the encrypted PGM image to verify that everything works.

*** Deliverable: Include the encrypted images in your report.**

*** Deliverable: Upload encrypted PGM images to the "Lab1 Dropoff" folder on Blackboard.**

***Task 5*:** Make note of any interesting features, especially potential shortcomings, of SIMON based on inspection of the encrypted PGM images.

*** Deliverable: Include these observations in your report.**