Cryptography on Hardware Platforms (WS 2023/24)

Assignment 1

(Only Task-1 of this assignment is released this week. The remaining tasks of this assignment will appear gradually.)

In Assignment-1, you will implement a public-key cryptography design following hardware/software codesign. The assignment consists of several tasks starting with Task-1.

Total points for Assignment-1 is 60 including 15 points for the individual oral defense.

Review class notes on modular arithmetic and Vivado tutorial before starting the assignment.

Task-1: Hardware implementation of a 64-bit modular multiplier (10 points)

Soft deadline: 12th November

You are given the 64-bit prime p = 18434813901432784897. Implement a modular multiplier circuit entirely in hardware that multiplies two elements $a, b \in F_p$ and produces the result $c \in F_p$ such that $c = a*b*R^{-1} \bmod p$ where $R=2^{64}$. The modular reduction must be performed using the *Montgomery method*.

The input/output ports of the module for this modular multiplier are defined as follows.

```
module modular_multiplier(clk, mul_ina, mul_inb, mul_start, mul_out, mul_done);
input clk;
input [63:0] mul_ina, mul_inb;  // Two 64 bit inputs a and b
input mul_start;                // When this signal is 1, the multiplication starts.
output [63:0] mul_out;          // Result of the modular multiplication is available in this port.
output mul_done;                // This signal becomes 1 when the multiplication ends.
```

Simulate the module with several inputs to verify the functional correctness of your modular reduction circuit.

Synthesize and implement the module using Vivado 2020.2 to obtain area and clock frequency reports. For the synthesis and implementation, the FPGA board must be set to Pynq-z2 in Vivado 2020.2.

The modular multiplier should obtain over 100 MHz clock frequency as the overall public-key processor will be clocked at 100 MHz in the FPGA.

Optimization hints: Generally it is not the case that a hardware design obtains high speed as well as very small low area. You can consider optimizing the modular reduction circuit for speed or area.

Bit-parallel architectures offer high throughput e.g., one modular reduction per cycle. Whereas, sequential architectures offer small area by using resource sharing between components. In both cases, pipeline registers play an important role in obtaining good clock frequency. As the modular multiplier performs several multiplications, try to optimally use DSP multipliers for obtaining a good design.

About optimizations, there is a great quote by a great person: "*If I had eight hours to chop down a tree, I'd spend six hours sharpening my axe.*" – so, plan your design well before writing the code.

We provide a Python model of Montgomery reduction operation for prime *p* on course webpage.

Task-2 will be announced later.

Submission guidelines

- The soft deadline for submitting your Task-1 is 12th November.
- Upload the Vivado project to the git repository of your team by the deadline.
- Upload your short one-page report to the git repository of your team by the deadline. The report should have:
    - Explanation of your design strategy.
    - Optimizations and design decisions that you made.
    - Implementation results of your design (resources and clock frequency).
    - Latency and throughput of your design.

Marking scheme

15 points are reserved for the individual oral defense of Assignment 1.

Task 1 (10 points)

- You get 7 points if the modular multiplier is functionally correct and the code is synthesizable in Vivado.

- You get 0 to 3 points based on how well your implementation of Task 1 is optimized in terms of resource requirements or speed or both.