Navaneeth Garakahalli
ECE 646 – Fall 2013

FPGA Implementation of Photon

Motivation

Photon is a family of lightweight hash functions designed primarily for hardware implementation. To cover a wide range of applications, multiple flavors/variants have been proposed. Photon has been implemented in an ASIC and the performance & security for different flavors/variants of Photon have been reported. I intend to put forth an implementation using FPGAs with high Throughput/Area ratio. This could possibly be the first FPGA implementation of this hash function.

Design tools

I intend to approach this design in Verilog. I would be using Altera Quartus II for my synthesis, ModelSim for simulation and Athena for benchmarking. Additionally, I intend to use Quincy for software simulation.

Design Source

A good source of information about Photon is the homepage, as indicated on the course webpage: https://sites.google.com/site/photonhashfunction/home

The website consists of the original paper published and C-implementation of the hash function. It also consists of some test vectors that can be used for testing the design post hardware implementation. More test vectors, if necessary, can be generated using the C implementation.

Assumptions

The Photon hash function has different flavors/variants of implementations based on the bitsize of the hash output and the input-output bitrates. I intend to implement Photon-256/32/32; output hash value of 256 bits, input & output bit rates of 32bits/cycle.

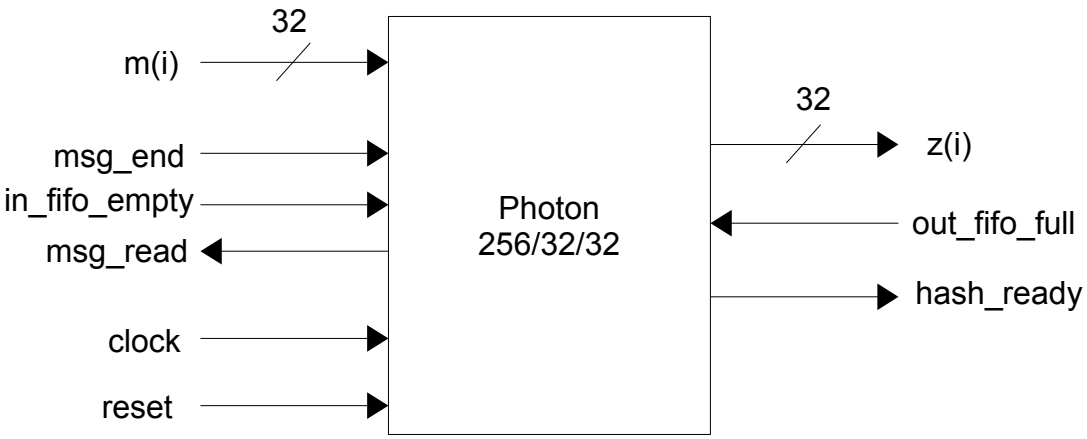
Hash functions, by definition, will take an input of arbitrary length and the output is a fixed size; in this case, 256 bits. I assume the input is provided to the system in block size of 32 bits/cycle. I also assume that the the blocks are provided with the necessary padding.

The implementation would be targeting Cyclone VI and Cyclone V family of FPGAs from Altera. Also, I will try to achieve maximum Throughput/Area ratio.

Interface

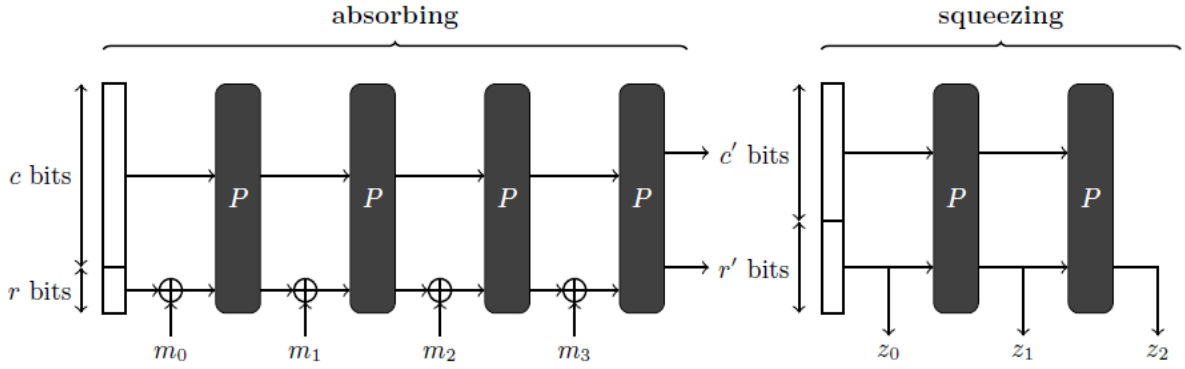
The interface consists of the output of hash function (256 bits wide in 32 bit blocks), message input (in 32 bit blocks).

The control signals include msg_read (to indicate the start of reading message from input fifo), msg_end (to indicate end of message input), in_fifo_empty (to indicate the input fifo is empty or not), hash_ready (to indicate the start/stop of hash output) and out_fifo_full (to indicate the output fifo is full).

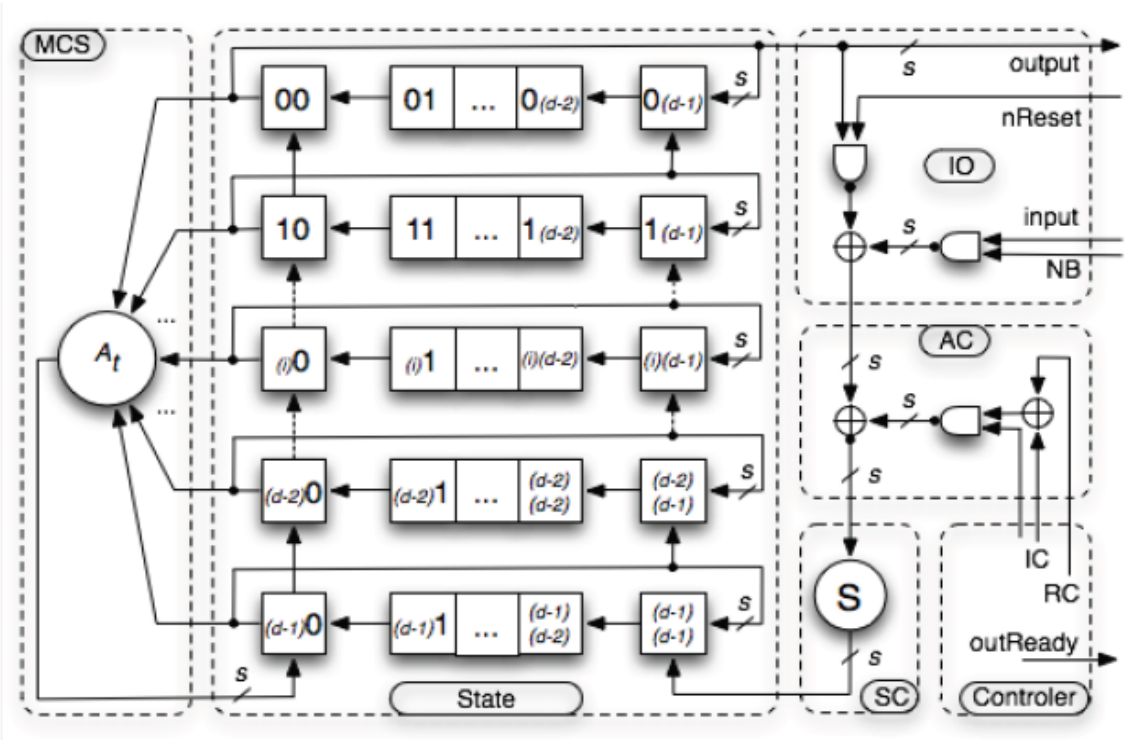


Previous implementation

The published paper consists of a Block diagram for the hash function.



The paper also provides the block diagram for serial implementation in hardware.



The literature provides results of previous implementations in ASIC. It also provides the performance of different hash functions in terms of throughput and area in Gate Equivalents (GE). Hence this could be the first implementation targeting FPGA.

Testing Performance and Functionality

Test vectors can be generated using the C-implementation.

Simulation tools will be used to determine the highest clock frequency at which the system can run. It will also be used to determine the number of logic elements used and also the usage of inbuilt memory. The tools will also be used to perform power analysis.

Finally the design could be modified to take in multiple blocks of message in a clock cycle and generate the output hash value in a single clock cycle.

The implementation would be limited to software simulation and would not involve verification using a prototyping board.

Schedule

09/25 – Specification
09/30 – Literature study
10/10 – Block diagram of Datapath
10/15 – Verilog code of datapath, testbench
10/20 – Synthesis and post synthesis simulation
10/30 – ASM chart of controller and its verilog code.
11/21 – Top level design and implementation in Cyclone IV & Cyclone V
Modifying design to maximize Throughput/Area ratio
12/01 – Benchmarking & Preparing deliverables

Literature list

Hash function specification
Paper: 'The PHOTON Family of Lightweight Hash Functions'
Authors: Jian Guo, Thomas Peyrin and Axel Poschmann
https://sites.google.com/site/photonhashfunction/downloads/photon-full.pdf
https://sites.google.com/site/photonhashfunction/design

Software implementation
https://sites.google.com/site/photonhashfunction/downloads/Reference-Implementation.zip

Test vectors
https://sites.google.com/site/photonhashfunction/downloads/testvectors.zip

AES
http://en.wikipedia.org/wiki/Advanced_Encryption_Standard

Sponge function
http://sponge.noekeon.org/

Tentative Table of Contents in Project Report

- Introduction
  - Motivation

- Photon Hash function
  - Variants
  - Permutations
  - Extended sponge framework

- Block diagrams
  - Top level design
  - Datapath
  - Controller

- Hardware implementation
  - Assumptions
  - Tools
  - Target FPGA
  - Optimization target

- Results
  - Throughput
  - Area
  - Power analysis
  - FPGA resources

- Conclusion
  - Comparison of implementations in different FPGAs
  - Future work

- References