Half-fast

A Bitcoin Miner for the FPGA

Overview

- Objectives and Motivation
- Bitcoin
- System Overview
- Hardware
- Software
- Challenges and Difficulties
- Lessons learned

Objectives and Motivation

- Build a Bitcoin miner on a FPGA board
- Mine block data from Bitcoin Network
- Parallelization

Bitcoin

- Bitcoin is an open source payment system based extensively on cryptographic hash functions
- Mining solves the problem of double spending through verifying transactions
- Transactions are public, but have no personal information
- Proof-of-work and mining pool

Proof-of-work

- Based on SHA-256
- Must find a number which added to a hashed header will fit a certain number of zeros (difficulty) by incrementing a number called the nonce
- Hashes change drastically with a tiny modification, turning it into a very complex problem

Mining pool

The mining pool is a process where multiple clients contribute to the solving of a block and share the rewards

Work is organized by leader. Block data is sent to miners to attempt to solve

Mining Algorithm

B = Block of Transactions

D = Difficulty (part of B)

1) Construct/Modify B
2) If SHA256(SHA256(B, nonce)) < D End
3) nonce++; Goto 1

SHA256 Algorithm

Message input M

Divide M into 512-bit chunks, pad if necessary

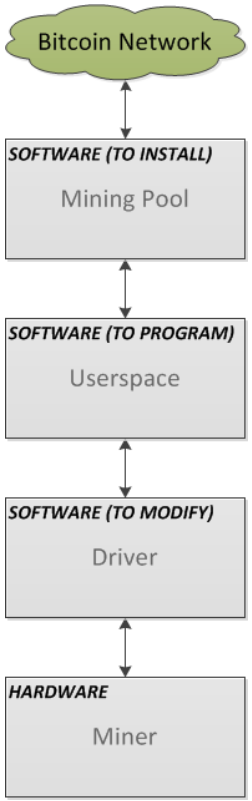
For each chunk Mi

    Compression(Mi) //bitwise shifting and rotation

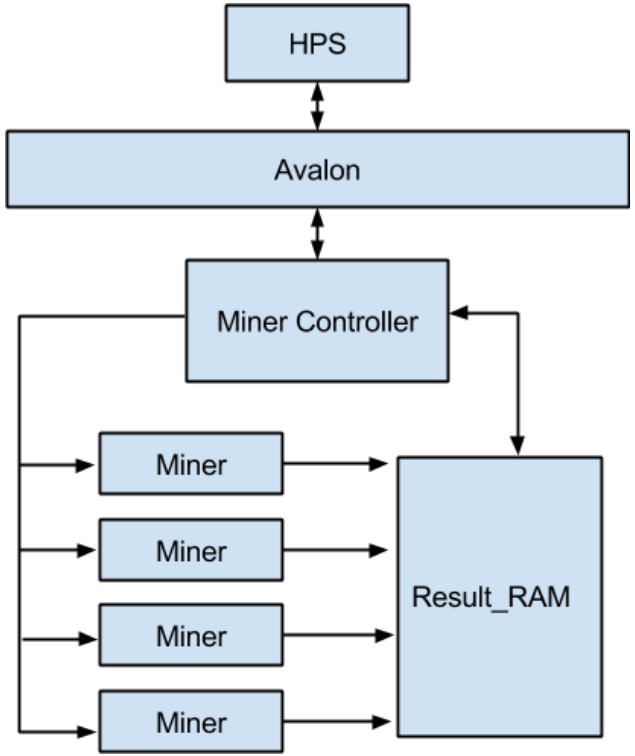
    Accumulate into registers h0, h1, … h7

hash = {h0, h1, h2, …, h7}

System overview

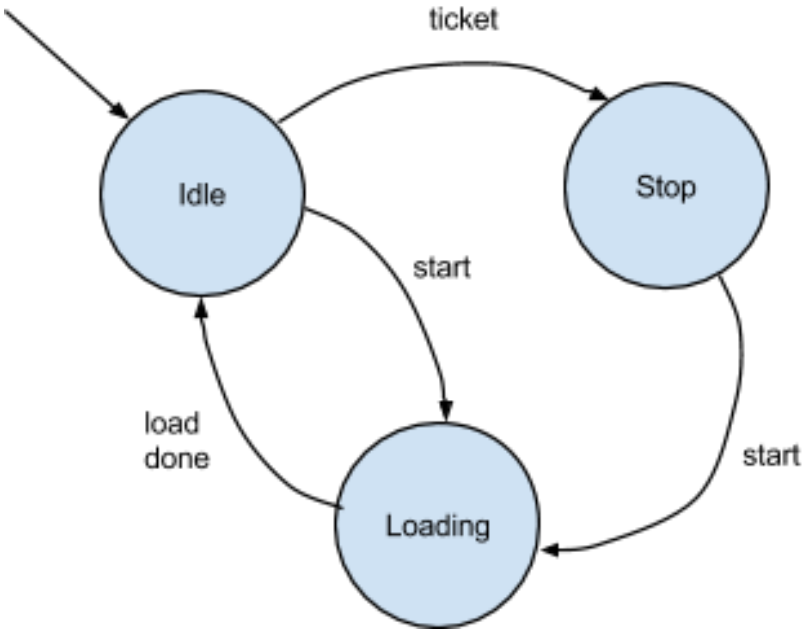


System overview



Hardware implementation

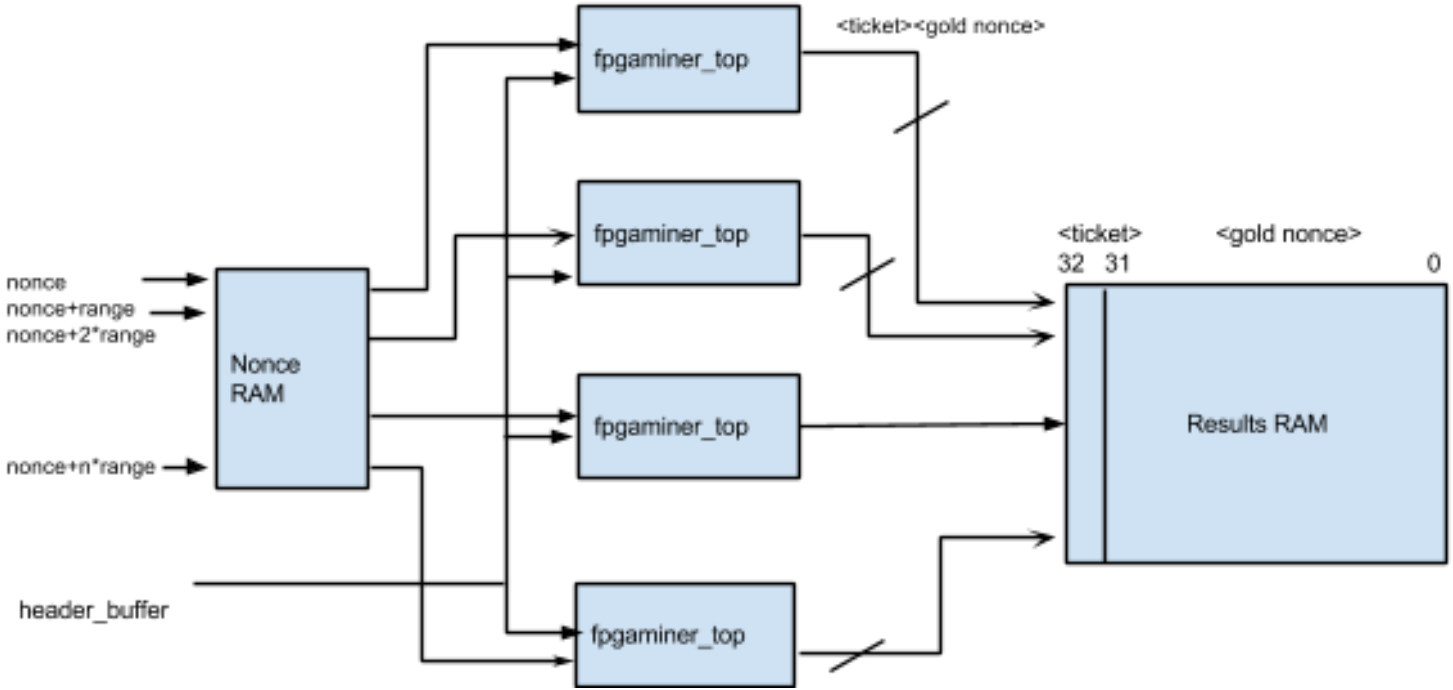
FSM



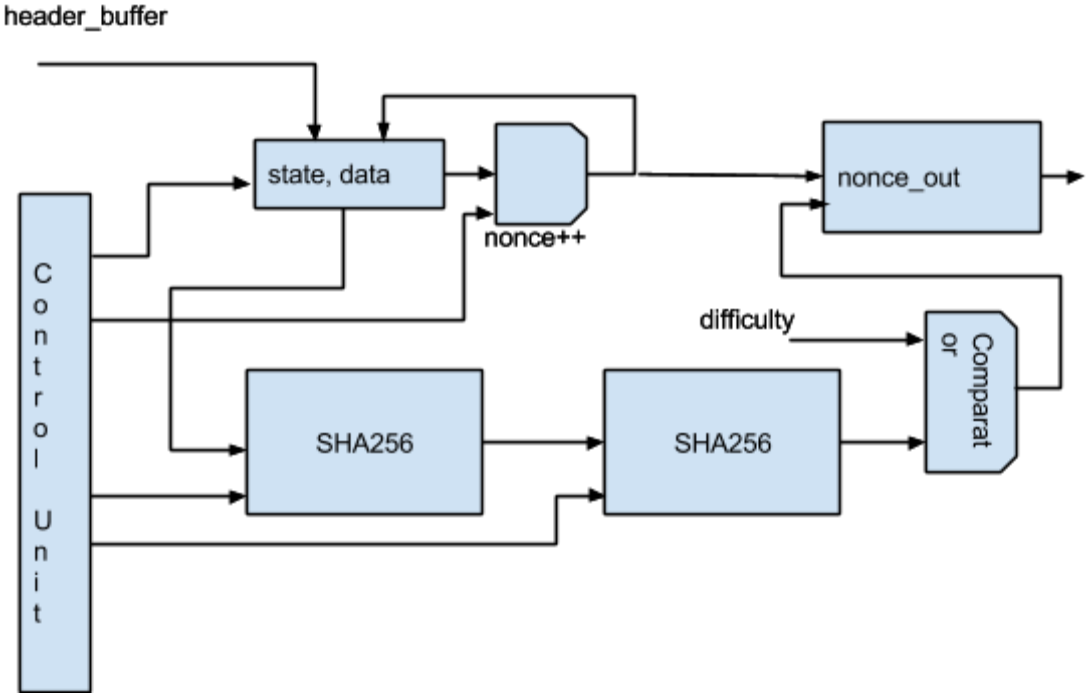
Memory Map Interface

input:	output:
clk reset write read chipselect writedata[7:0] address[7:0]	readdata[7:0]

Miner Top


fpgaminer_top
fpgaminer_top
fpgaminer_top
fpgaminer_top
<ticket><gold nonce>
<ticket>
<gold nonce>
32 31
0
Results RAM
nonce
nonce+range
nonce+2*range
nonce+n*range
Nonce
RAM
header_buffer

FPGA Miner*



*Used an Open Source Miner. Modified it for our interface

https://github.com/gardintrapp/Open-Source-FPGA-Bitcoin-Miner

SHA256

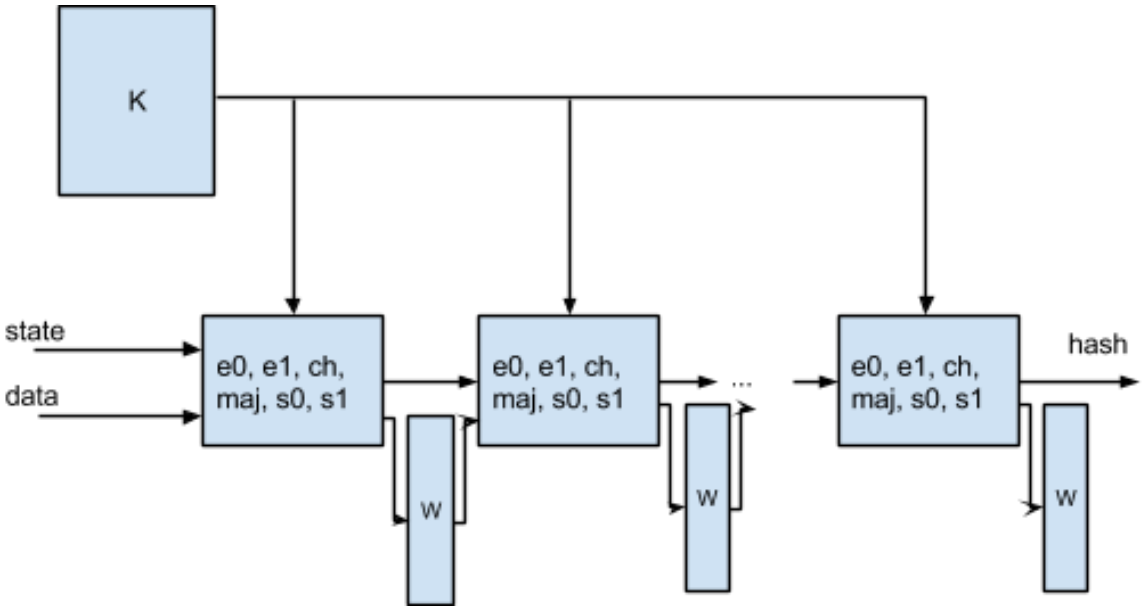
e0, e1, ch, maj, s0, s1 - bitwise operations

LOOP parameter determines how many “digester” blocks are instantiated

Big LOOP = less space, slower

Small LOOP = more space, faster

K is array of constant values



This is the Compression function unrolled

Software implementation

getwork.c

- Userspace program written to facilitate communication between Mining pool and our FPGA miner
- Creates a getwork request to mining pool
- Sends the work down to the hardware with IOCTL calls defined in modified vga_led.c/h
- Separate threads reads and listens for solved work from fpga and new work from the mining pool

Challenges and difficulties

- Debugging hardware logic
- Writing scalable Verilog code
- Bookkeeping data and Simulating
- Learning the Bitcoin system

Lessons learned

- Be more thorough with initial planning/design process
- Simulate/Test carefully and thoroughly at each step of hardware implementation (ModelSim and System Console). Use scripts
- Start from Lab3 skeleton code
- Work on hardware and software in parallel
- Begin hardware software integration as early in development as possible

Half-fast: a Bitcoin miner for the FPGA

Thank you!