AsicBoost -
A Speedup for Bitcoin Mining

Dr. Timo Hanke
March 29, 2016
rev. 2

1 Introduction

AsicBoost is a method to speed up Bitcoin mining by a factor of approximately 20% across all existing mining hardware.

The *AsicBoost* method is based on a new way to process work items inside and outside of the Bitcoin mining ASIC. It involves a new design of the SHA 256 hash-engines (inside the ASIC) and an additional pre-processing step as part of the mining software (outside the ASIC). The result is a performance improvement of up to 20% achieved through a reduction of gate count on the silicon. The purpose of this paper is to present the idea behind the method and to describe the information flow in implementations of *AsicBoost*.

The hash-engine design required for *AsicBoost* is compatible with design philosophies such as “rolled cores” and “fully pipelined cores”. The performance gains can be achieved on top of all other optimizations regarding timing, pipelining, path balancing, custom cell and full-custom designs.

Through gate count reduction on the silicon *AsicBoost* improves two essential Bitcoin mining cost metrics simultaneously and by a similar factor: the energy consumption (Joule per Gh) and the system cost ($ per Gh/s). With the system cost being proportional to the capital expenses of a Bitcoin mine and the energy consumption being proportional to its operating expenses, *AsicBoost* reduces the total cost per bitcoin mined by approximately 20%. For the Bitcoin mines of the future *AsicBoost* will make all the difference between a profitable and an unprofitable mine.

The *AsicBoost* method was invented by Timo Hanke and Sergio Demian Lerner and is patent-pending.

http://asicboost.com/
asicboost@gmail.com

2 Preliminaries on Bitcoin Mining

2.1 SHA 256

The SHA 256 function of a message is calculated based on dividing the message into *chunks* of 64 bytes each, and processing the chunks consecutively through a state machine. The final state after having processed all chunks yields the SHA digest of the original message.

During processing, each chunk is run through a *message expander* function which produces a corresponding *message schedule* of 64 words[1]. The message schedule is then fed into a *compressor* function which changes its internal state in the process of consuming the message schedule in rounds, one word at a time. The compressor's state after having processed all message schedules derived from all chunks of the original message turns into the final hash digest of the original message.

2.2 The Bitcoin block header

In Bitcoin mining the message to be hashed is the *block header*. A Bitcoin block header is 80 bytes long and is divided into two chunks as follows, the second chunk being padded to a length of 64 bytes:

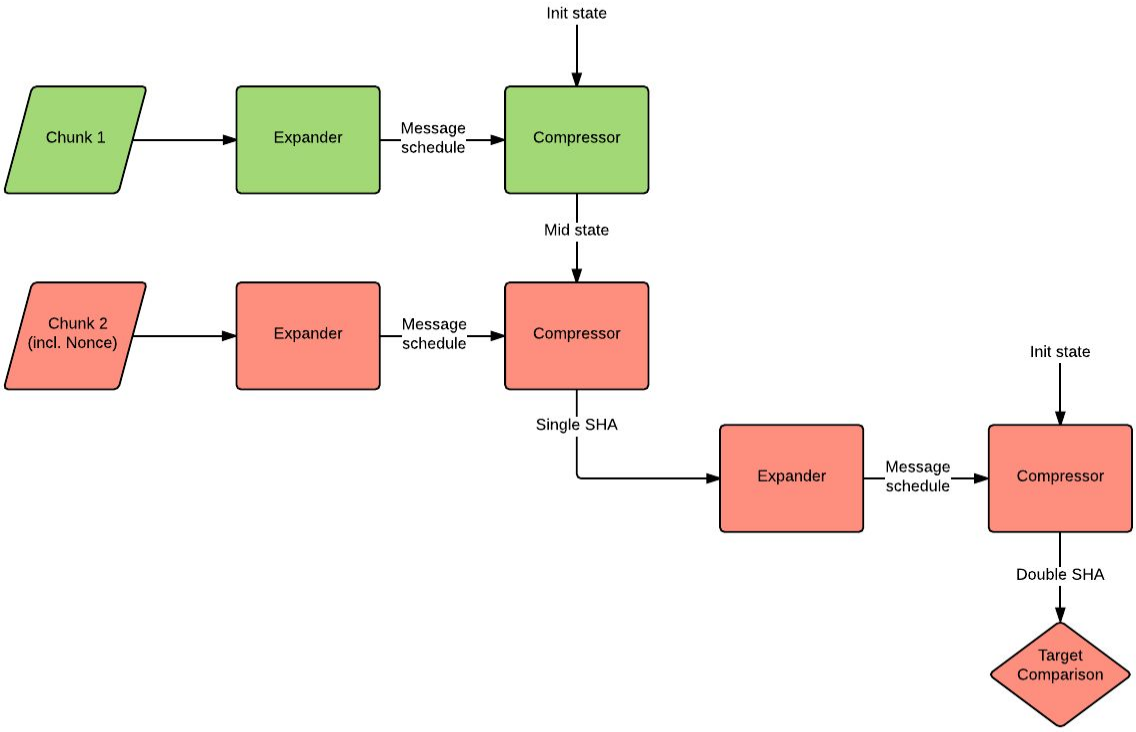
<table>
<tr><td colspan="3">Chunk 1</td><td colspan="5">Chunk 2</td></tr>
<tr><td colspan="6">Block header</td><td rowspan="3">Padding</td></tr>
<tr><td colspan="5">Block header candidate</td><td rowspan="3">Nonce</td></tr>
<tr><td rowspan="2">Version</td><td rowspan="2">Previous hash</td><td colspan="2">Merkle root</td><td rowspan="2">Time stamp</td><td rowspan="2">Bits (difficulty)</td></tr>
<tr><td>Head</td><td>Tail</td></tr>
<tr><td>4 bytes</td><td>32 bytes</td><td>28 bytes</td><td>4 bytes</td><td>4 bytes</td><td>4 bytes</td><td>4 bytes</td><td>48 bytes</td></tr>
<tr><td colspan="3"></td><td colspan="3">Message[2]</td><td></td></tr>
</table>

[1] A *word* is 4 bytes.
[2]The first 12 bytes of Chunk 2 are called *Message* for the purpose of this document.

http://asicboost.com/
asicboost@gmail.com

2.3 Bitcoin's mining function

Bitcoin's mining function processes block headers with a double SHA, and the resulting information flow for both chunks through the double SHA looks like this:



The resulting double SHA digest is compared against the target range based on the current network difficulty, and if a match is found then the block header (including the *Nonce* that produced the match) is broadcast to the Bitcoin network.

2.4 Scanning the Nonce range

The Bitcoin mining process selects a *block header candidate*, loops over the whole Nonce range and filters out the Nonces that produce a double SHA digest matching the target. When the Nonce range is exhausted a new block header candidate is selected.

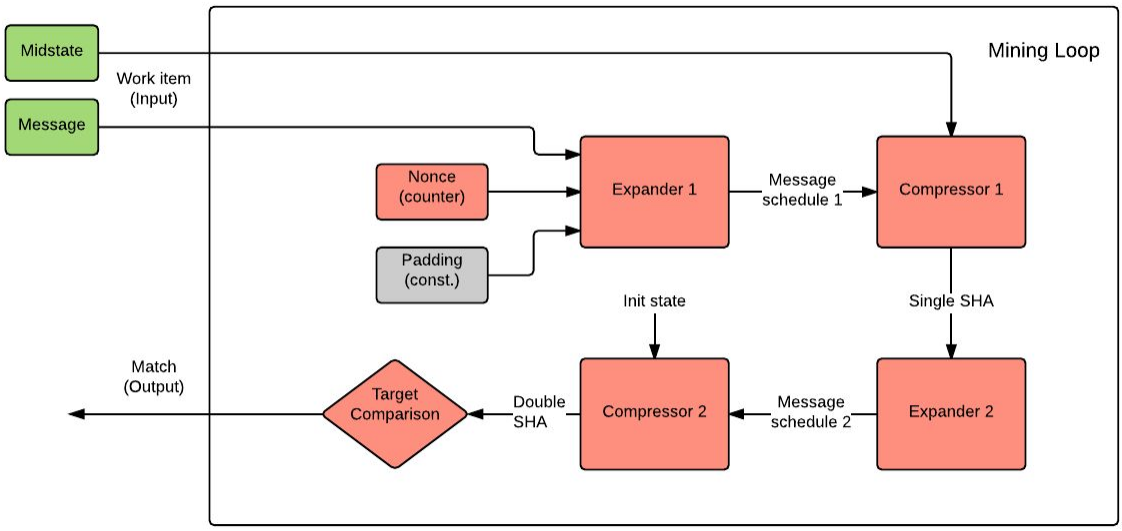
The part of the computation that is repeated in the loop over the Nonce range is depicted in red in the above diagram. This part depends on the Nonce. The green part is not repeated in the loop as it depends only on the block header candidate, not on the Nonce.

The loop over the Nonce range starts out with this information, together called a *Work item*:

- Mid state
- Message

The Mid state is the output of the green compressor function. The Message is the part of Chunk 2 that depends on the block header candidate. The Message excludes the Nonce (which is selected inside the loop) and the padding (which is constant).

In light of this, the following diagram shows the core part of the computation involved in Bitcoin mining, or the *Bitcoin mining loop*:



All Bitcoin mining ASICs only process the Bitcoin mining loop internally. The work items are pre-computed and passed to the ASIC from outside.

2.5 Building Work items

Each work item is constructed from a new block header candidate which in turn is constructed from a new *Merkle root*. Since the Merkle root spans both chunks of the block header, updating the Merkle root affects both components of the work item: Midstate (depending on chunk 1) *and* Message (depending on chunk 2). Therefore, it is usually not the case that different work items share either of the two components. This is the reason why the information in *Message Schedule 1* (see diagram) cannot be re-used across the processing of multiple work items.

3 AsicBoost

3.1 Gain through colliding work items

AsicBoost achieves its performance gain by highly re-using the Message Schedule 1 across multiple work items. As a result, almost the entire computational work required for the Expander 1 function can be saved. This leaves only the Compressor 1, Expander 2 and Compressor 2

functions to be processed inside the Mining Loop. Since each of the four functions have similar complexity, and *AsicBoost* eliminates one of them from the Mining Loop, it can save up to one quarter of the total computational work per work item.

In order to be able to re-use Message Schedule 1 across multiple work items, *AsicBoost* generates many block header candidates that all share a common Message part. In other words, the many block header candidates differ from each other only in chunk 1. The work items derived from these block header candidates then all share a common Message component and differ only in the Midstate component. We speak of them as *colliding work items* because they collide in the Message component. A set of colliding work items allows an *AsicBoost*-enabled chip to re-use Message Schedule 1 across the processing of all of them.

3.2 AsicBoost chip design

There are several hash-engine layouts that allow to re-use the Message Schedule 1. For example, different hashing cores on the chip can be loaded with colliding work items simultaneously and share the output of a single Expander 1 block. The optimal implementation will depend on many low-level factors of the original hashing core design.

We consult on the best strategy to adopt AsicBoost into your existing chip design in regards to on-chip work distribution, nonce handling, prevention of idle times or lag, layout, timing issues, clock distribution.

3.3 AsicBoost software

There are several ways to produce block header candidates that differ only in chunk 1. For example, many merkle roots can be calculated and filtered based on their last 4 bytes until sufficiently many are found that collide in their last 4 bytes. *AsicBoost* provides several methods to calculate merkle tree roots efficiently and finding collisions quickly. The optimal implementation will depend on many factors of the whole mining environment, starting at the mining pool software and down to the on-board microcontroller.

We consult on the best strategy to adopt AsicBoost into your existing system design and mining infrastructure for anything from stand-alone mining devices to centrally controlled multi-petahash installations. This includes getting your mining pool protocol ready for *AsicBoost*-enabled devices.