AION FPGA Data Path

Earl Mai
emai@epicblockchain.io
July 23rd, 2018

Revision 0.1

AION FPGA Data Path

Scope

This document describes the data path of the Equihash RTL design as data is read and written to memory.

Memory Elements

There are 2 types of data being written and read from memory and described below in detail. All access to memory is 256bit wide for both the XOR data and Pair data.

XOR data (256 bit)

The XOR data is initially generated by the blake2b block and written into memory. In stage 0, the indices used to generate the data are stored in the upper 32 bits and padded with an identifier to allow detecting that this is an indices.

XOR data bit assignments

bits	***Description***
32	Address (blake2b, pair)
14	Unused
210	XOR data

Address blake2b bit assignments

bits	***Description***
10	0x3FF Identifier
22	blake2b Index

Address pair bit assignments

bits	***Description***
32	64bit Aligned Memory Address to a Pair

Pair data (64 bit)

The pair data represents a binary tree containing pointers to the sub nodes used in formingof the pair. The 64 bit Pair data is packed 4 to a memory access allowing efficient memory savings and keeping with the 256 bit width accesses to and from memory.

Pair data bit assignments

bits	*Description*
32	Upper Address
32	Lower Address

The special case arises for leaf nodes when the Pair data contains 2 blake2b indices and can be detected by checking for the 0x3FF identifier for both the Upper and Lower address regions.

Memory Buffers

There are 3 memory buffers used in the design and referred to as MEM_BUF0, MEM_BUF1 and MEM_BUF2. Each region is allocated 128MB of physical memory requiring each core to use 384MB of memory. These parameters can be changed as needed and note that MEM_BUF2 is 64bit aligned.

Memory Map

The map below specifics which MEM_BUF each block read and writes to during each stage.

stage	*blake2b write*	*radix read*	*radix write*	*collision read*	*collision write*	*pair read/write*
0	**0**					
1		**0**	**0**	**0**	**1**	**2**
2		**1**	**1**	**1**	**0**	**2**
3		**0**	**0**	**0**	**1**	**2**
4		**1**	**1**	**1**	**0**	**2**
5		**0**	**0**	**0**	**1**	**2**
6		**1**	**1**	**1**	**0**	**2**
7		**0**	**0**	**0**	**1**	**2**
8		**1**	**1**	**1**	**0**	**2**
9		**1**	**1**	**0**		**2**

*Note during the radix operation, scratch space is used in every pass. For the (210,9) implementation and 4bit radix sort, 6 passes are required thus the resultant sorted data is located in the original location.

**Note in Stage 9, there are on average 2 binary search tree operations occurring to find the solution set.

Memory Access Count

For worst case analysis there are $L = 2^{22}$ data elements that need to be processed in each stage.

Stage 0: 1 *L* accesses
Stage 1: 7 *L* (6L radix, 1L collision)
Stage 2: 7 *L* (6L radix, 1L collision)
Stage 3: 7 *L* (6L radix, 1L collision)
Stage 4: 7 *L* (6L radix, 1L collision)
Stage 5: 7 *L* (6L radix, 1L collision)
Stage 6: 7 *L* (6L radix, 1L collision)
Stage 7: 7 *L* (6L radix, 1L collision)
Stage 8: 7 *L* (6L radix, 1L collision)
Stage 9: 7 *L* (6L radix, 1L collision)

Total Access = 2^6 *L accesses*