# Method

All communication with the hardware is done using UDP on Gb Ethernet. All UDP packets in both directions have payloads of 1,000 bytes. For each packet that is sent by the host CPU, an equal sized packet is returned by our design. Our encoding allows each character to be represented in to two bits. The first frame sent by the host CPU contains configuration information including what set of comparisons to perform, the maximum Hamming distance, the number of subsequent frames, and the actual tags for which to search.

The top level block diagram of the design is shown in Figure 1. We use UDP communication to communicate with the host CPU. For this, we reused the Ethernet reference design from MP1. To simplify the design, the AddressSwap and lower modules were removed and a fixed UDP return header used instead. Given the fixed packet lengths, using a fixed header is straightforward.



Figure : Top level of design

In addition to the Digital Clock Manager (DCM) providing a 125 MHz clock for the Ethernet module, a second DCM is required to drive our design at 25 MHz. A reset line is held high until the DCMs are locked preventing activity before the clocks have stabilized.

The Driver module (Figure 2) is our abstraction layer between the communication layer and the rest of the design. It encompasses the comparators, tag reversal gates, and tag complementing gates.



Figure : Detail of Driver unit

The Incoming FSM strips off the Ethernet header and loads the payload into the In FIFO. For each byte read from the In FIFO, the Cmp Driver stores a result byte into the Outgoing FIFO. The FIFOs are used to separate clock domains and also provide a predictable buffer, alleviating the need for flow control. The Outgoing FSM constructs the outgoing frame by presenting a UDP header to the linklayer and then streaming the contents of the Out FIFO.

Note that the Incoming and Outgoing FSMs operate in parallel. On startup, the Incoming FSM loads the configuration data in the first frame and genome data in the second frame. For each subsequent frame, it shifts genome data through the In FIFO while the Outgoing FSM shifts the results from the Outgoing FIFO. Since the bottleneck is calculation, as opposed to communication, this simultaneous shifting of genome data in and results data out ensures that the calculation stage remains busy.

The comparison driver shown in Figure 3 loads the configuration settings and genomes into the comparison units. In the first frame, the Compare FSM loads the Maximum Hamming Distance and tags into the Cmp Units. The Compare FSM streams the genome in the second and subsequent frames to the Cmp Units and controls the Units’ comparisons against the tags.



Figure : Detail of comparison driver

The reset locks the Compare FSM, Shift Unit, and Cmp Units until “in fifo ready” signals the arrival of data. The FSM replies by setting the “in fifo rd en” to high and starts reading data from the “in fifo dout” signal.

The first frame programs the FSM to control whether the comparisons of tags are compared in the forward or reverse directions, and whether the tags and k-mer should be used as-is or complemented. The FSM reads the Hamming Distance and programs that into all of the Cmp Units. The Hamming Distance specifies how many mismatches may be tolerated and still declare a match between the tag and the k-mer. Perfect matches are not always expected due to small genetic drift or errors in sequencing.

The FSM then loads tags into the Cmp Units. This design makes use of the large degree of parallelism provided by the FPGA. Each Cmp Unit holds a unique tag. As each tag arrives in the frame, the FSM shifts the tag into the Shift Unit. After the tag is loaded, the FSM places the tag number (1..N) on the “index” signal and activates the “load tag” signal. The Cmp Unit corresponding to that tag number loads the tag into its tag register.

After the tags are loaded, the FSM streams the genome through the Shift Unit. Each k-mer in the genome is compared by each of the Cmp Units. The FSM uses the forward, backward, and complement signals to drive the comparisons in the Cmp Units. Each Cmp Unit supplies a match/no-match result. All of these results are reduced to a final 2-bit result for that position in the genome. (How is this result constructed and why is it two bits?)

The replication of the Compare Units is shown in more detail in Figure 4. This is the second and largest level of parallelism in the design. Replicating N Units allows N tags to be compared at one time. A program running on a standard processor would have to loop over all the tags for each k-mer in the genome. The design presented here uses a Compare Unit to represent each step in that loop.



Figure : Replication of comparison unit

The replicated Comparison Unit that compares a tag with each k-mer in the genome is shown in Figure 5.



Figure : Detail of Comparison Unit

The Comparison Unit features four identical pipelines that reduce a comparison of the tag against the current k-mer into a single match/no-match Boolean. The pipeline counts the number of bit positions that differ between the tag and k-mer. If that count is less than or equal to the Hamming Distance, the tag is considered to match the k-mer.

As each Comparison Unit is generated it stores its index. The Load Tag signal combined with the Index signal control when the Nth tag is loaded into the Nth Unit’s Tag register. The Load Max HD signal loads the Hamming Distance into the Max HD register.

Two pipelines use the normal tag, while the other two use the reverse complement of the tag. Similarly, two pipelines use the k-mer, while two use the reverse of the k-mer. This combination of pipelines allows comparison of the tag, either normal or complemented, and the direction of the tag, either forward or reverse. The forward, reverse, and reverse-complement signals control which pipeline’s output is used for the result.

Each pipeline begins with an xor stage to find the number of bits that do not match. The mash stage or’s the even bits with the odd bits. The Ones Counter stage uses a tree structure of parallel adders to count the number of ones which is the actual Hamming distance of the k-mer to the tag. This value is compared to the maximum allowed Hamming Distance to declare if this tag matches.