EE5332, Mapping Signal Processing Algorithms to Architecture, Project Review
Genome Sequence Alignment on FPGAs

Group:

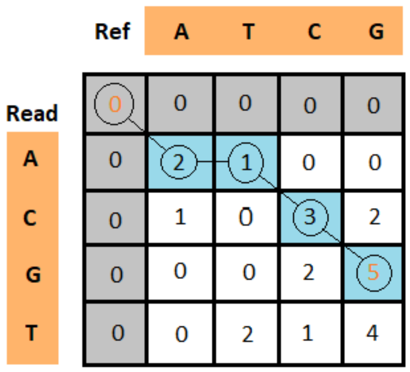
- **Sai Gautham Ravipati (EE19B053)**
- **Vishnu Varma V (EE19B059)**
- **Shashank Nag (EE19B118)**

Smith-Waterman Algorithm

The Smith-Waterman algorithm is one of the critical algorithms used in DNA sequencing technology. It is used in the alignment of two genome sequences (reference and read sequence). The goal is to align these two sequences effectively, considering the effects of errors such as mismatch, insertion, and deletion (insertion and deletion are termed as gap errors). When such errors occur they have to be penalised and when the corresponding letter of the sequence matches then it has to be rewarded. Based on this a scoring matrix is constructed where the alignment scores across both the sequences are recursively calculated based on matches and errors and updated in the matrix. The scoring strategy used is as follows:

$$H_{i,j} = \max \begin{cases} H_{i-1,j-1} + S(i,\ j) & : \text{(Mis)Match} \\ H_{i-1,j} \ - W_1 & : \text{Insertion} \\ H_{i,j-1} \ - W_1 & : \text{Deletion} \\ 0 & : \text{Baseline} \end{cases}$$

Where for the following example, S(i,j) is 2 for the match, and -2 for a mismatch. If $\text{Ref}_i = \text{Read}_j$ then it is a match else it is a mismatch. W_1 is chosen as 1 for this example. The first row and column of the score matrix are all initialised to 0, and then all the other scores for the cells are recursively calculated and updated.



Read \ Ref		A	T	C	G
	0	0	0	0	0
A	0	2	1	0	0
C	0	1	0	3	2
G	0	0	0	2	5
T	0	0	2	1	4

Figure. Final Scoring Matrix

Let us consider the computation of the score for H(3,4), the maximum value among the three neighbours must be taken, i.e, left: 3, top: 3, upper left: 3 + 2 = 5 (+2 because G-G matching). So max(3,3,5) gives 5 which is the score associated with that cell. Similarly, all the score values are computed. Now backtracking is performed, which is tracing back the highest scores among the top left and upper left neighbours starting from the highest score of the whole score matrix. Starting from the highest value of 5 comparing the neighbours the highest score among the neighbours is 3 so it is traced. Then corresponding to 3 the highest score is 1 - it is traced back, and repeated recursively until getting a 0. If the score comes from the upper left then it is either a match or a mismatch, if it is from the left score then it is a deletion which results in adding the "-" character into the read sequence, if it is from the top score then it is insertion which results in adding "-" character into the reference sequence. Backtracking the example above gives the following alignment:

Reference - **A T C G**
Read - **A - C G** ("-" in the read sequence corresponds to insertion)

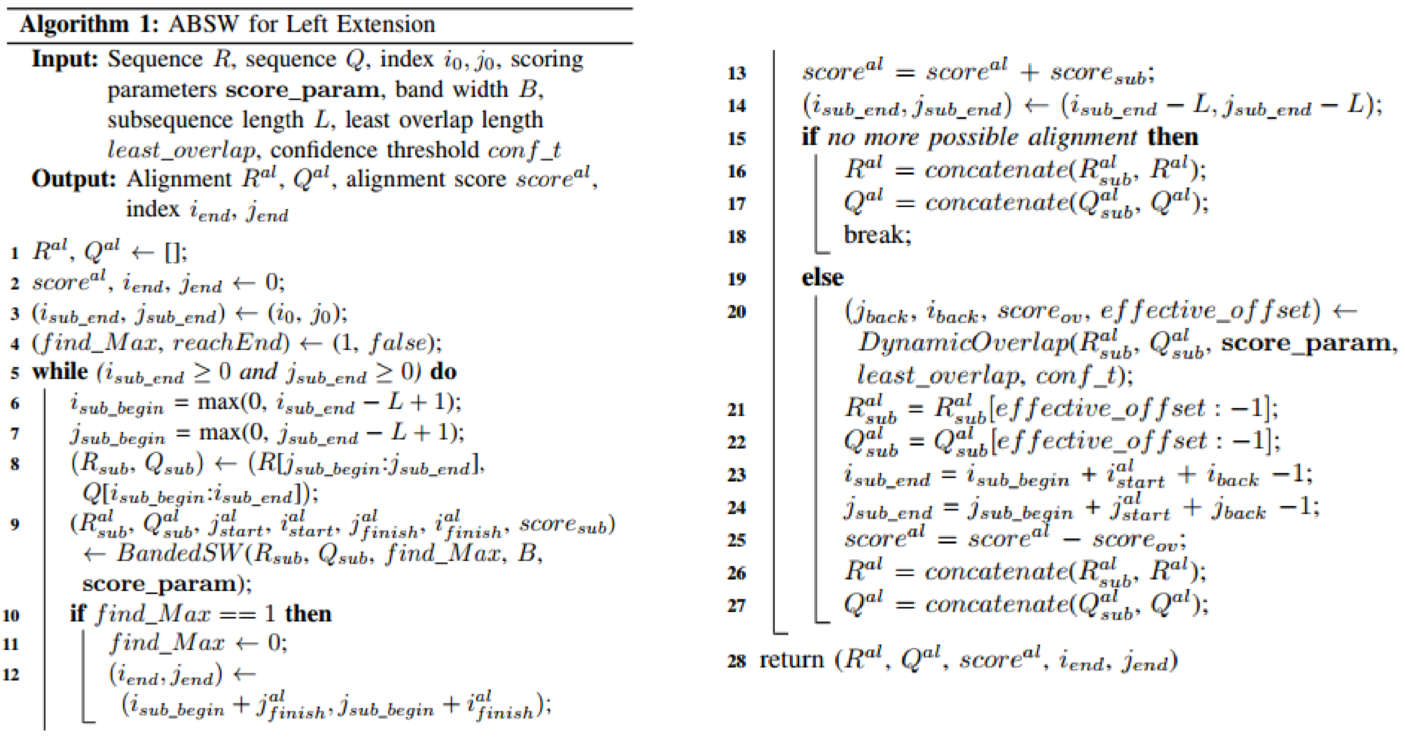
Banded Smith-Waterman

In general, Smith-Waterman is used to select the best match among the reads after passing it through BWT (Burrows-Wheeler Transform) which is an efficient way of compressing the sequences that remove most of the bad candidates. So the optimal alignment path is usually close to the diagonal of the matrix. So instead of computing for the whole matrix band around the diagonal can be considered for computation which is called Banded Smith-Waterman. Relevant papers to the above algorithm have been reviewed and the main ones of them have been summarised below.

Adapted Banded Smith-Waterman (ABSW) Algorithm, Liao et. al.

Liao et al., in their paper, Adapted Banded Smith-Waterman Algorithm for long reads and its hardware accelerators, propose an adaptively banded Smith-Waterman Algorithm that is hardware-compatible. The banded Smith-Waterman algorithm computes the cell scores only in a band about the diagonal, in contrast to other algorithms like GACT that take into account the entire space. This is based on the observation that the optimal alignment paths lie close to the diagonal. The paper claims that this approach doesn't result in any significant loss in accuracy while having substantial gains in terms of speedup.

From the hardware perspective, most of the previous works on the banded Smith-Waterman involve performing the score computing part on the hardware, and the traceback part is performed on the software. This is one of the first works where the traceback part is also performed on the hardware. This work adopts a seed and extends paradigm, where initially, in the seed phase, short subsequences (seeds) are aligned, followed by the extension phase where the read alignment is extended around the bases. The seed phase is performed on software in this work, while the extended phase is on hardware.



Algorithm 1: ABSW for Left Extension

Input: Sequence R, sequence Q, index i_0, j_0, scoring parameters **score_param**, band width B, subsequence length L, least overlap length $least\_overlap$, confidence threshold $conf\_t$
Output: Alignment R^{al}, Q^{al}, alignment score $score^{al}$, index i_{end}, j_{end}

```
$R^{al}, Q^{al} \leftarrow []$;
$score^{al}, i_{end}, j_{end} \leftarrow 0$;
$(i_{sub\_end}, j_{sub\_end}) \leftarrow (i_0, j_0)$;
$(find\_Max, reachEnd) \leftarrow (1, false)$;
while $(i_{sub\_end} \geq 0$ and $j_{sub\_end} \geq 0)$ do
    $i_{sub\_begin} = \max(0, i_{sub\_end} - L + 1)$;
    $j_{sub\_begin} = \max(0, j_{sub\_end} - L + 1)$;
    $(R_{sub}, Q_{sub}) \leftarrow (R[j_{sub\_begin}:j_{sub\_end}], Q[i_{sub\_begin}:i_{sub\_end}])$;
    $(R^{al}_{sub}, Q^{al}_{sub}, j^{al}_{start}, i^{al}_{start}, j^{al}_{finish}, i^{al}_{finish}, score_{sub}) \leftarrow BandedSW(R_{sub}, Q_{sub}, find\_Max, B,$ score_param$)$;
    if $find\_Max == 1$ then
        $find\_Max \leftarrow 0$;
        $(i_{end}, j_{end}) \leftarrow (i_{sub\_begin} + j^{al}_{finish}, j_{sub\_begin} + i^{al}_{finish})$;
    $score^{al} = score^{al} + score_{sub}$;
    $(i_{sub\_end}, j_{sub\_end}) \leftarrow (i_{sub\_end} - L, j_{sub\_end} - L)$;
    if no more possible alignment then
        $R^{al} = concatenate(R^{al}_{sub}, R^{al})$;
        $Q^{al} = concatenate(Q^{al}_{sub}, Q^{al})$;
        break;
    else
        $(j_{back}, i_{back}, score_{ov}, effective\_offset) \leftarrow DynamicOverlap(R^{al}_{sub}, Q^{al}_{sub},$ score_param$, least\_overlap, conf\_t)$;
        $R^{al}_{sub} = R^{al}_{sub}[effective\_offset : -1]$;
        $Q^{al}_{sub} = Q^{al}_{sub}[effective\_offset : -1]$;
        $i_{sub\_end} = i_{sub\_begin} + i^{al}_{start} + i_{back} - 1$;
        $j_{sub\_end} = j_{sub\_begin} + j^{al}_{start} + j_{back} - 1$;
        $score^{al} = score^{al} - score_{ov}$;
        $R^{al} = concatenate(R^{al}_{sub}, R^{al})$;
        $Q^{al} = concatenate(Q^{al}_{sub}, Q^{al})$;
return $(R^{al}, Q^{al}, score^{al}, i_{end}, j_{end})$
```

Figure 1. Algorithm for Sequence Alignment

In the indicated algorithm (Fig. 1), BandedSW refers to the compute-intensive alignment algorithm, for which a hardware accelerator is proposed by the authors. As can be seen from the equation for computing H_{ij}, the score cells along the anti-diagonal can be processed independently, and hence, a processing element (PE) array-based architecture has been proposed for parallelism, as in Figure 2 (the PE architecture is as shown in Fig. 3). For implementing the algorithm on two sequences of length L, with the bandwidth of B, this architecture uses B PEs - thus allowing the B anti-diagonal scores to be processed parallely. The PEs complete calculating the scores of the matrix within (2L - 1) cycles, with shift registers being used to feed in the shifted sequences to the PEs for computing scores. Note that there is a data dependency involved amongst the PEs, which is dealt with by the custom-built data dependency module. The relative positions for the traceback phase are stored in B BRAMs, referred to as the traceback memory. In the traceback phase, these BRAMs are read and they are converted into the actual pointers by taking into account the corresponding PE id. The total number of cycles required to perform the hardware implementation of BandedSW for this architecture is given by:

$$C = B + 2(L - 1) + 2 + 1.125L_{al}$$

where L_{al} is the length of alignment (~L). For the same number of PEs, this is only about 40% of the number of cycles required for the GACT accelerator, which computes the Smith Waterman algorithm over the entire space instead of a band. The BandedSW hardware and PE are represented in Fig. 2 and Fig. 3, note that PE shall collapse for the case when o is same as e. (All images taken from source paper)

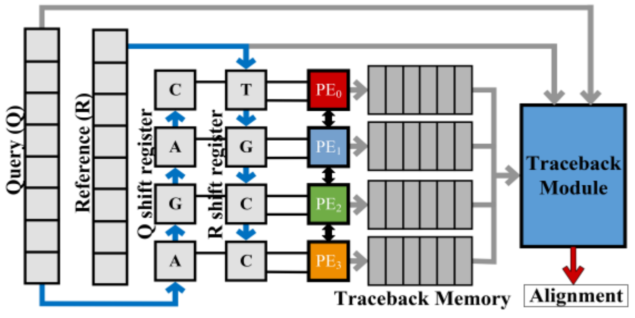


FIgure 2. BandedSW Hardware

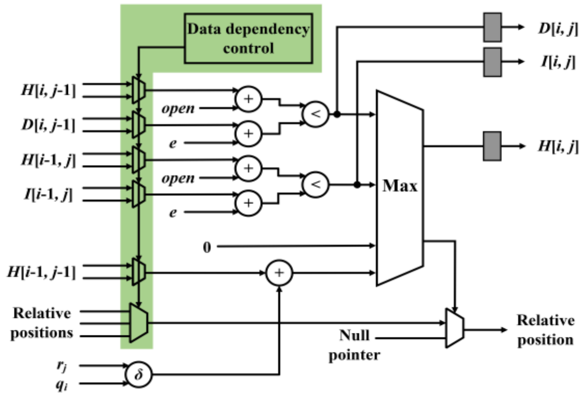


Figure 3. Processing element used in BandedSW

PipeBSW: A Two-Stage Pipeline Structure for Banded Smith-Waterman Algorithm on FPGA, Li et. al.

PipeBSW is an accelerator design that reduces memory movement between the host CPU and an FPGA and tries to implement both scoring and traceback phases on hardware in a pipelined fashion. Although scoring is the most time-consuming stage (claimed to be ~90%), the movement of intermediate representations between the host CPU and FPGA between stages may pose a problem when the length of the reads increases. The paper proposes a hardware back-tracking module (BTU) generally implemented on software in the literature to mitigate the same. For the band's construction over the diagonal, a PE-based architecture is adopted, then operated on by the BTU. Both of them are connected in a pipeline, which also provides a possibility of hardware re-use. Further, a lookahead calculation technique is adopted to reduce the number of cycles to complete calculations.

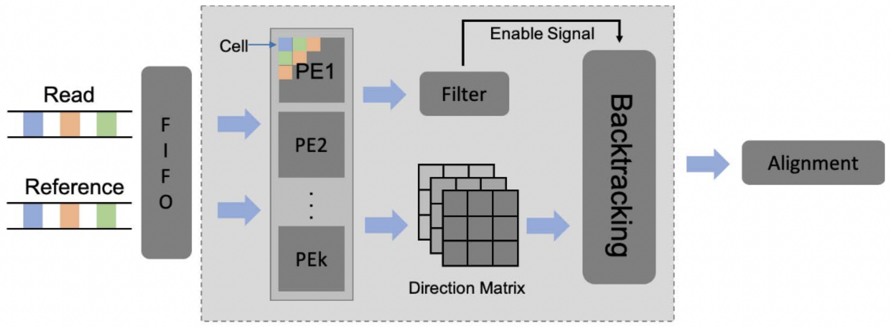


Figure 1. Architecture of PipeBSW

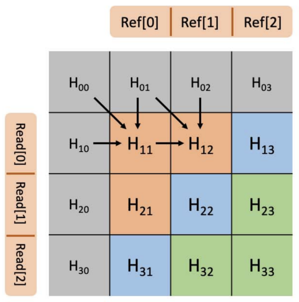


Figure 2. Cell (3 x 3) used in PE

The workflow of PipeBSW is represented in the (Fig. 1) from the source paper, read and reference sequences are sent to PEs after being sliced out, and each PE computes the scoring matrix parallelly. A PE is composed of thirteen 3 x 3 cells (systolic array structures), to process 36 x 36 elements. The structure of PE is attached (Fig. 2) where elements processed in the same cycle are in the same colour. Here, in this case, H_{12} depends on H_{11} but performing a lookahead computation, potentially both of them could be computed in the same cycle. So computation that would take 5 cycles in a general systolic array computation has been reduced to 3 cycles. In this approach 3 positions (H_{12}, H_{21}, H_{33}) require lookahead computation, and specifically, they are chosen for a balance in the critical path length. The 13 cells in a PE are re-used for different cycles, where elements in the anti-diagonal direction score synchronously, and 25 positions at the end of a PE forming an 'L' region are considered to form a band while back-tracing. If the path goes beyond this band it means that total modifications are greater than 12bp for a 36bp read, which means the read is a low-quality candidate. The maximum element in the L region is tracked at the end of each PE which shall act as a starting point for back-tracing. Several such PEs are stacked (Fig. 3) to form a diagonal band with an intersection across their L regions, and backtracking effectively happens in these regions.

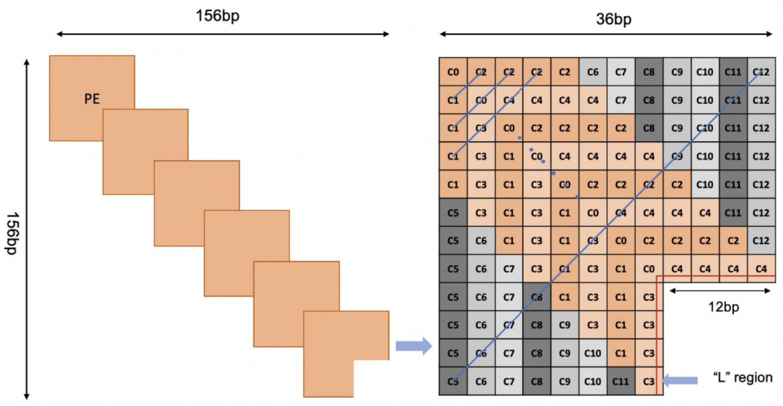


Figure 3. Overlap of PEs to form a band and Scheduling of cells in a PE

For the Backtracing, we need to store direction information and it could lead to mismatch, match, insertion or deletion, which are to be encoded as 2-bits and since we are traversing only across the band, we should encode this information for elements only in this band, by this we are replacing a scoring matrix with a direction matrix and it is advantageous when we move to the case of large reads. The Backtracing module reads the direction information at a given location and appends the 2-bits to the alignment path which is to be correlated with the reference and read to get the generated sequence. The number of cycles to complete the backtracing shall depend on the length of the aligned string which could go to 36 in the worst case with mismatches or 24 if all of them match. For the backtracing we need serial computation, so having 6 BTU units like the 6 PEs is a waste of resources, hence the paper adopts a 2-stage pipeline structure with one stage being the scoring step and the other stage backtracing. The pipeline is shown in Fig. 4 and as seen it allows the usage of the same BTU module repeatedly, while at the same time allowing for the re-use of PEs as well. The only constraint to be satisfied is when the BTU begins backtracing the direction matrix of the same PE that should already have been generated. Although there have been several metrics provided in comparison to the previous paper it is not sufficiently rigorous, while the claimed throughput is 720.9 Mbps. (All Images taken from source paper)

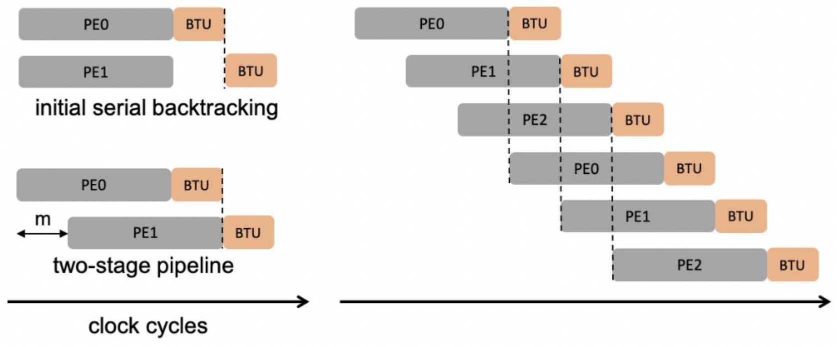


Figure 4. A pipelined implementation of stages

Other than these, there have been several other research works targeting genome sequence alignment with alternate algorithms. “Darwin” is one such interesting work, where instead of relying on a banded region about the diagonal (as in Banded Smith Waterman), it employs a novel filtration strategy called D-SOFT to reduce the search space. It also uses a GACT algorithm based hardware algorithm for alignments that restricts to use of constant memory for traceback. “SLIDER” is another interesting work, which also involves creating a pre-alignment filter. Interestingly, this filter is also efficiently implementable on hardware and, at the same time, does not alter the alignment step in any way. Another work, the “Wavefront Alignment Algorithm”, is an entirely different algorithm that aims at achieving optimal alignment at a low execution time.

Since this field involves several different algorithms, each with its own set of optimisations and hardware implementations, it would be beyond our scope to review all of those. As such, we would be going ahead with the Banded-Smith Waterman algorithm, and hardware implementations based on those. We would be implementing a software baseline for the Adapted Banded Smith-Waterman algorithm (Liao et al.) and identify if the bottleneck is actually the BandedSW part as indicated in the paper. Based on it, we would implement a hardware accelerator for this part alone and interface it with a general-purpose processor, as proposed by (Liao et. al.). Subsequently, we shall try to implement the PipeBSW paper by (Li et. al.), which targets a pipelined implementation completely on hardware for extending to long reads. An extensive analysis of the three cases would be performed by benchmarking the results.

The exact details of these implementations and split up of work for the implementation and analysis aspects would be decided after a discussion of the same with the instructor.

References

[1] : Adaptively Banded Smith-Waterman Algorithm for Long Reads and Its Hardware Accelerator, *Liao et. al.*
[2] : PipeBSW: A Two-Stage Pipeline Structure for Banded Smith-Waterman Algorithm on FPGA, *Li et. al.*
[3] : Darwin: A Hardware-acceleration Framework for Genomic Sequence Alignment, *Turakhia et. al.*
[4] : SLIDER: Fast and Efficient Computation of Banded Sequence Alignment, *Alser et. al.*
[5] : Fast gap-affine pairwise alignment using the wavefront algorithm, *Marco-Sola et. al.*