BOSTON UNIVERSITY

FPGA Acceleration of DNA Sequence Mapping

Team: DNA Mappers

ADITYA NARAYAN AASTHA ANAND G SIVA PERUMAL

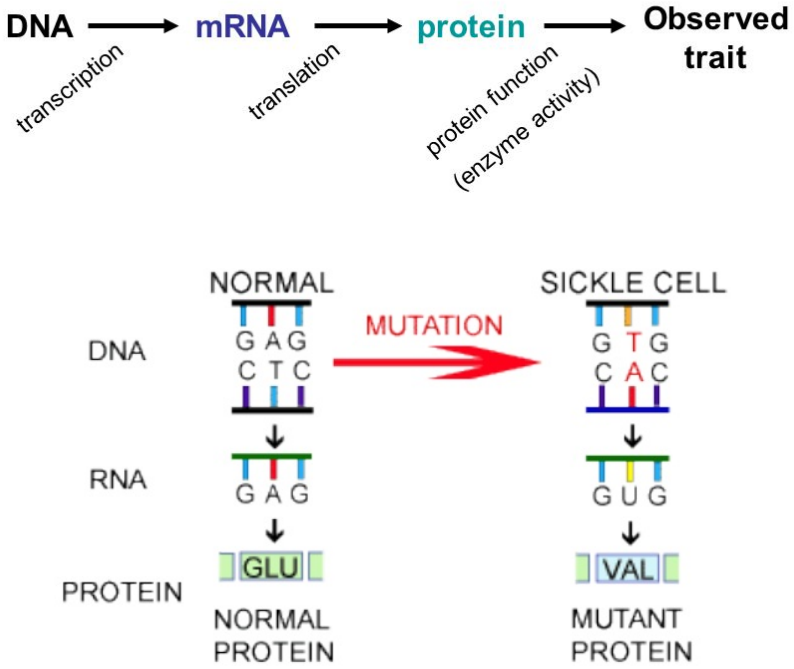
SARTHAK JAGETIA ANAND SHIVALKAR

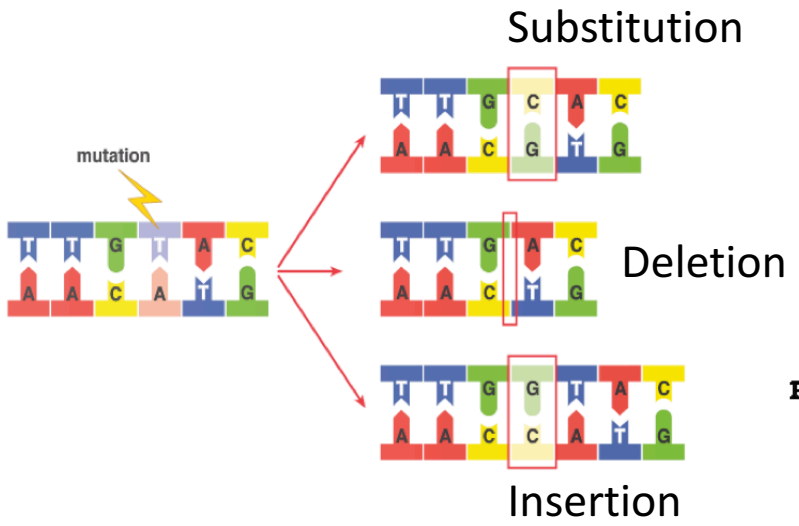
BOSTON UNIVERSITY

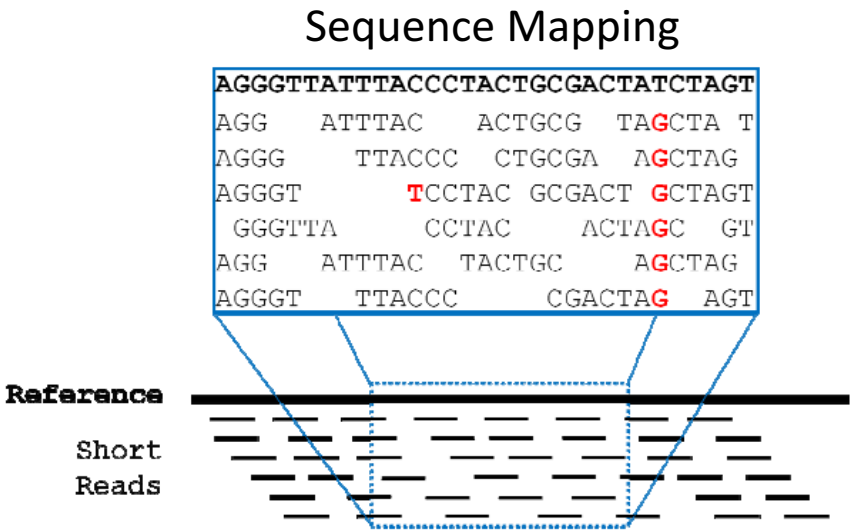
- **Background and Motivation**
- **High Level Diagram and Specifications**
- **Functional Description of Blocks**
- **Results and Analysis**
- **Successes and Failures**
- **Conclusion and Future Work**

Background

Central dogma of molecular biology

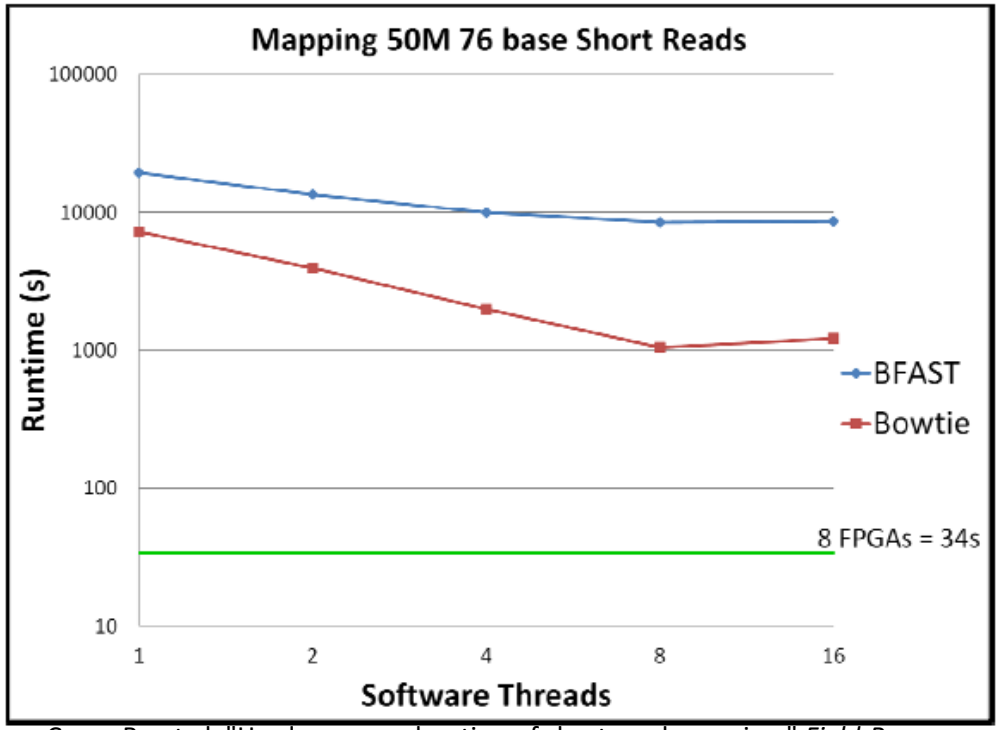






DNA Sequence Mapping on FPGA

- Short read mapping has traditionally been performed by software tools such as Bowtie, BWA , MAQ and BFAST running on a cluster of processors.
- Huge Computational burden due to comparison of >1 billion short reads against a very long reference genome
- Require 30-50 CPU days for complete mapping and alignment
- Can be significantly accelerated by the use of FPGAs and GPUs



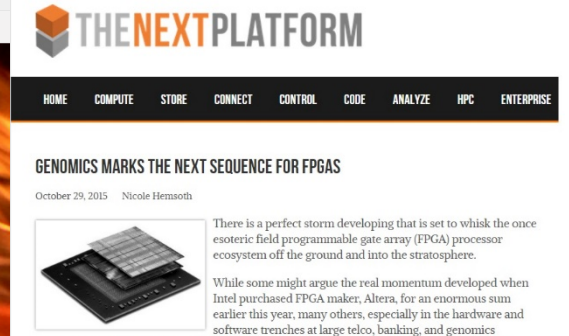
Olson, Corey B., et al. "Hardware acceleration of short read mapping." *Field-Programmable Custom Computing Machines (FCCM), 2012 IEEE 20th Annual International Symposium on*. IEEE, 2012.

Scope and Motivation

DRAGEN Bio-IT Platform

Enabling the Global Genomic Infrastructure

The DRAGEN (Dynamic Read Analysis for Genomics) Platform is based on the highly reconfigurable DRAGEN Bio-IT Processor. The DRAGEN Processor uses a field-programmable gate array (FPGA) to provide hardware-accelerated implementations of genome pipeline algorithms, such as BCL conversion, compression, mapping, alignment, sorting, duplicate marking and haplotype variant calling. The highly flexible DRAGEN platform allows Edico Genome to develop custom algorithms as well as refine and improve existing pipelines.


THENEXTPLATFORM

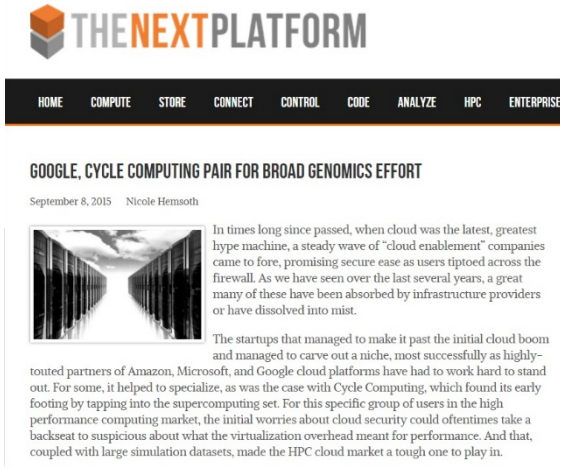
HOME COMPUTE STORE CONNECT CONTROL CODE ANALYZE HPC ENTERPRISE

GENOMICS MARKS THE NEXT SEQUENCE FOR FPGAS

October 29, 2015 Nicole Hemsoth

There is a perfect storm developing that is set to whisk the once esoteric field programmable gate array (FPGA) processor ecosystem off the ground and into the stratosphere.

While some might argue the real momentum developed when Intel purchased FPGA maker, Altera, for an enormous sum earlier this year, many others, especially in the hardware and software trenches at large telco, banking, and genomics companies, could explain how the impetus for an FPGA boom has been gathering force for far longer.


THENEXTPLATFORM

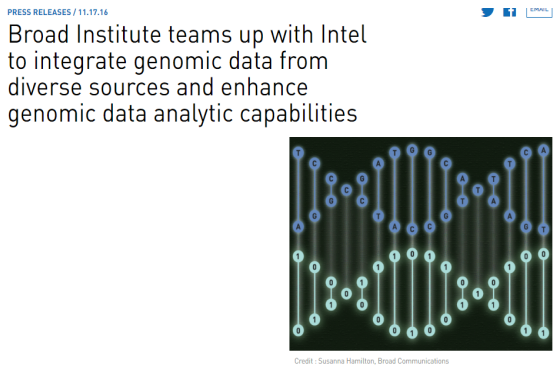
HOME COMPUTE STORE CONNECT CONTROL CODE ANALYZE HPC ENTERPRISE

GOOGLE, CYCLE COMPUTING PAIR FOR BROAD GENOMICS EFFORT

September 8, 2015 Nicole Hemsoth

In times long since passed, when cloud was the latest, greatest hype machine, a steady wave of "cloud enablement" companies came to fore, promising secure ease as users tiptoed across the firewall. As we have seen over the last several years, a great many of these have been absorbed by infrastructure providers or have dissolved into mist.

The startups that managed to make it past the initial cloud boom and managed to carve out a niche, most successfully as highly-touted partners of Amazon, Microsoft, and Google cloud platforms have had to work hard to stand out. For some, it helped to specialize, as was the case with Cycle Computing, which found its early footing by tapping into the supercomputing set. For this specific group of users in the high performance computing market, the initial worries about cloud security could oftentimes take a backseat to suspicious about what the virtualization overhead meant for performance. And that, coupled with large simulation datasets, made the HPC cloud market a tough one to play in.


PRESS RELEASES / 11.17.16

Broad Institute teams up with Intel to integrate genomic data from diverse sources and enhance genomic data analytic capabilities

Credit : Susanna Hamilton, Broad Communications

Cambridge, MA, November 17th, 2016

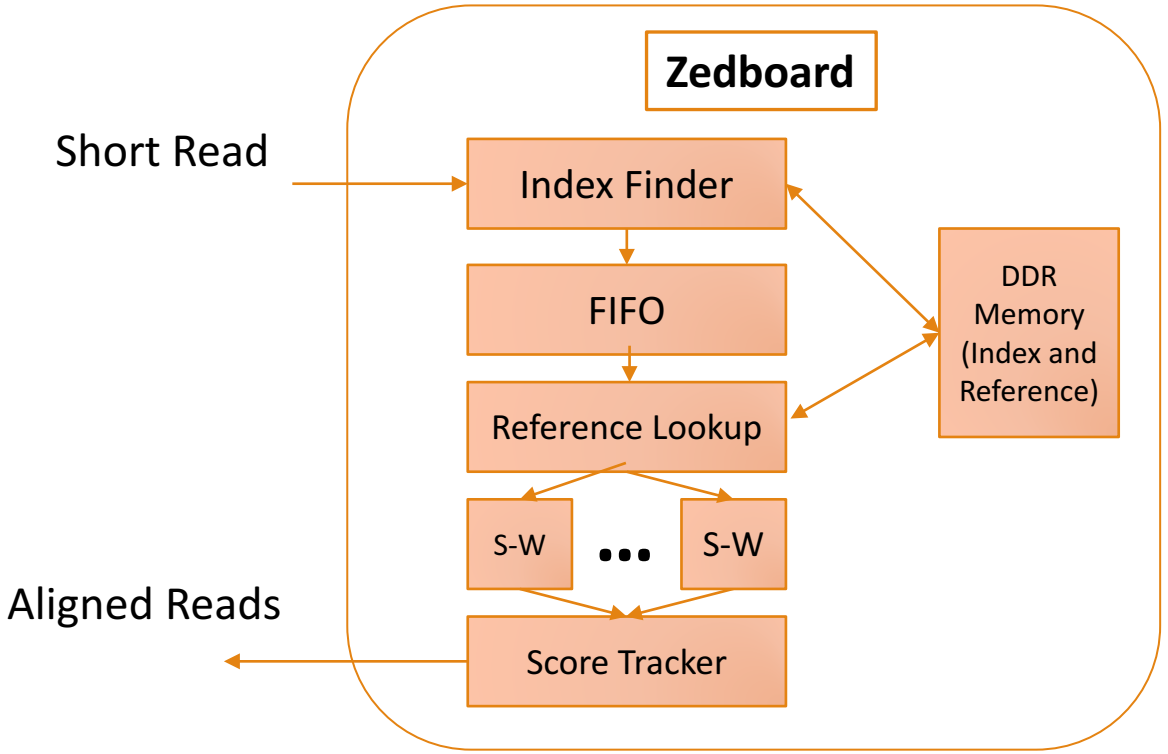
The Broad Institute of MIT and Harvard today announced a $25 million collaboration with Intel Corporation to scale researchers' ability to analyze massive amounts of genomic data from diverse sources worldwide.

Through a five-year collaboration, researchers and software engineers at the new Intel-Broad Center for Genomic Data Engineering will build, optimize, and widely share new tools and infrastructure that will help scientists integrate and process genomic data. The project aims to optimize best practices in hardware and software for genome analytics to make it possible to combine and use research data sets that reside on private, public,

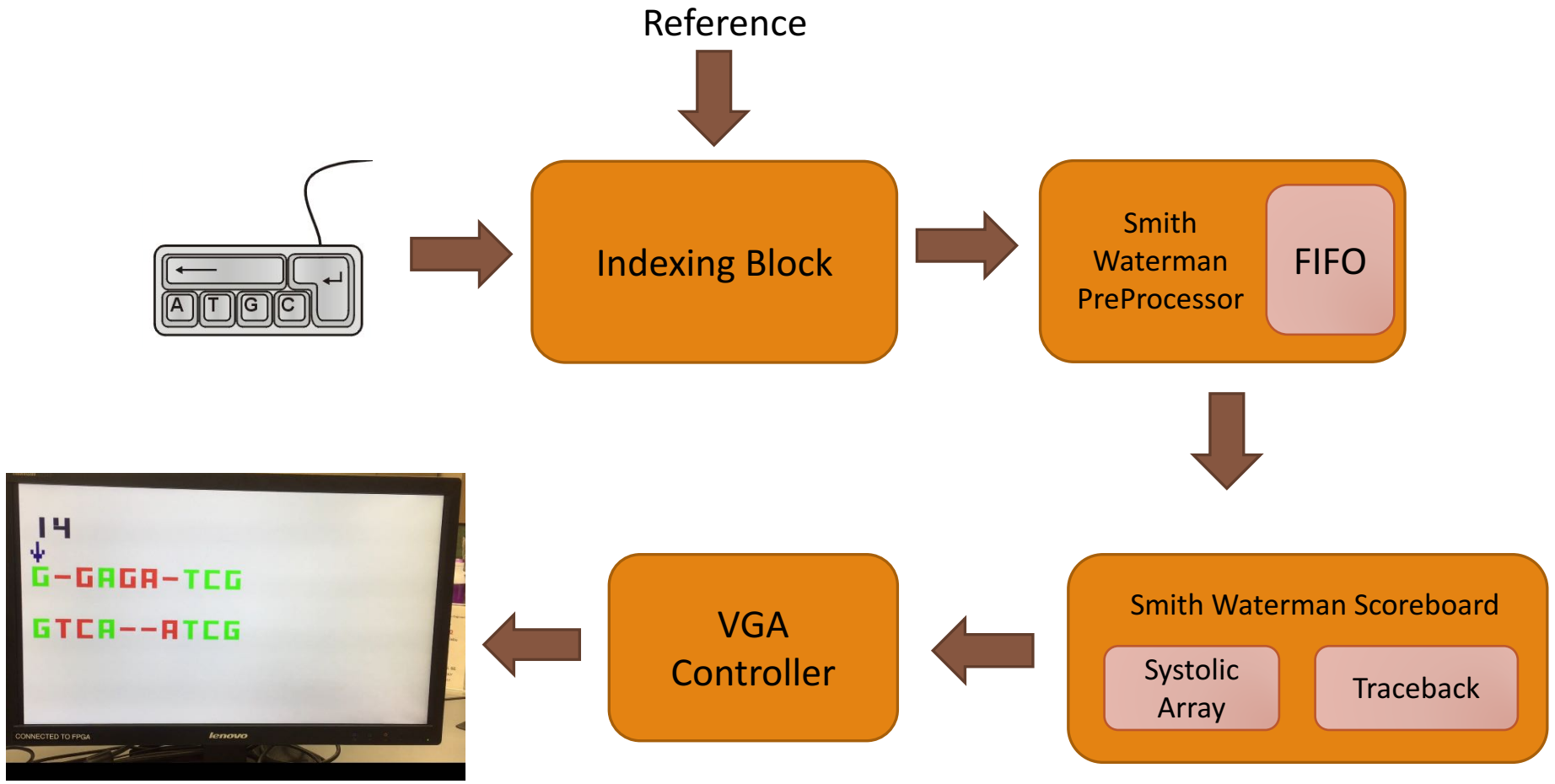
BOSTON UNIVERSITY

- Background and Motivation
- High Level Diagram and Specifications
- Functional Description of Blocks
- Results and Analysis
- Successes and Failures
- Conclusion and Future Work

Initial High Level Block Diagram



Final High Level Block Diagram

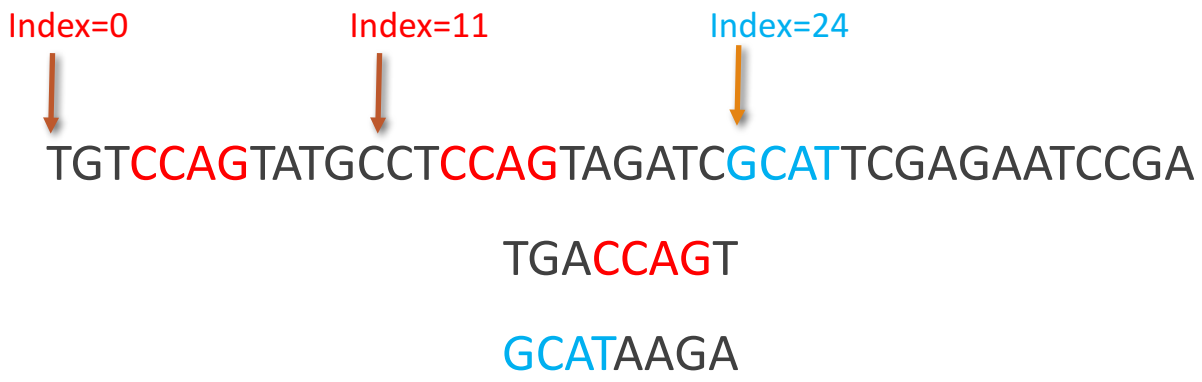


BOSTON UNIVERSITY

- Background and Motivation
- High Level Diagram and Specifications
- **Functional Description of Blocks**
- Results and Analysis
- Successes and Failures
- Conclusion and Future Work

Indexing

- Break down reference sequence into seeds (4 characters long) and store them in separate CAL tables
- Compare seeds from the input short reads with the reference seeds
- Identify all the indexes in the reference sequence where the short read can map with a high likelihood



Verilog Code Implementation

```
case(ref[count +: 2])
2'b00:
begin
mem_a[a_pointer] <= {ref[count  +:8] , count};
a_pointer <= a_pointer + 1;
end
2'b10:
begin
mem_c[c_pointer] <= {ref[count  +:8] , count};
c_pointer <= c_pointer + 1;
end
2'b01:
begin
mem_g[g_pointer] <= {ref[count  +:8] , count};
g_pointer <= g_pointer + 1;
end
default:
begin
mem_t[t_pointer] <= {ref[count  +:8] , count};
t_pointer <= t_pointer + 1;
end
endcase
count <= count + 2'b10;
end
else
begin
count <= count;
done <= 1'b1;
```

```
for (  l = 0;l <61; l = l+1)
begin
tempreg_a = mem_a[l];
if(short_read[countshortread +: 8]==tempreg_a[8 +:8])
begin
if(tempreg_a[7:0] - countshortread >= 0 & tempreg_a[7:0] - countshortread <= 84)
index = tempreg_a[7:0] - countshortread;
sequence = ref[index +: 16];
```

BOSTON UNIVERSITY

Smith Waterman Pre-Processor

- Generate sequence 1 from the reference sequence using the index
- Sequence 2 is the input short read
- Store the sequences 1 and 2 in a FIFO until Smith Waterman is ready to compute :
  - Smith Waterman takes n^2 clock cycles to compute
  - Indexing provides output every clock cycle until done

Local Sequence Alignment – Smith Waterman Algorithm

Why Smith Waterman?

- Commonly used for comparison of 2 strings
- The fully gapped algorithm provides accurate scoring, even in the presence of indels
- Search space is quadratic as opposed to exponential search space of other algorithms
- Can be implemented efficiently using 2D systolic arrays

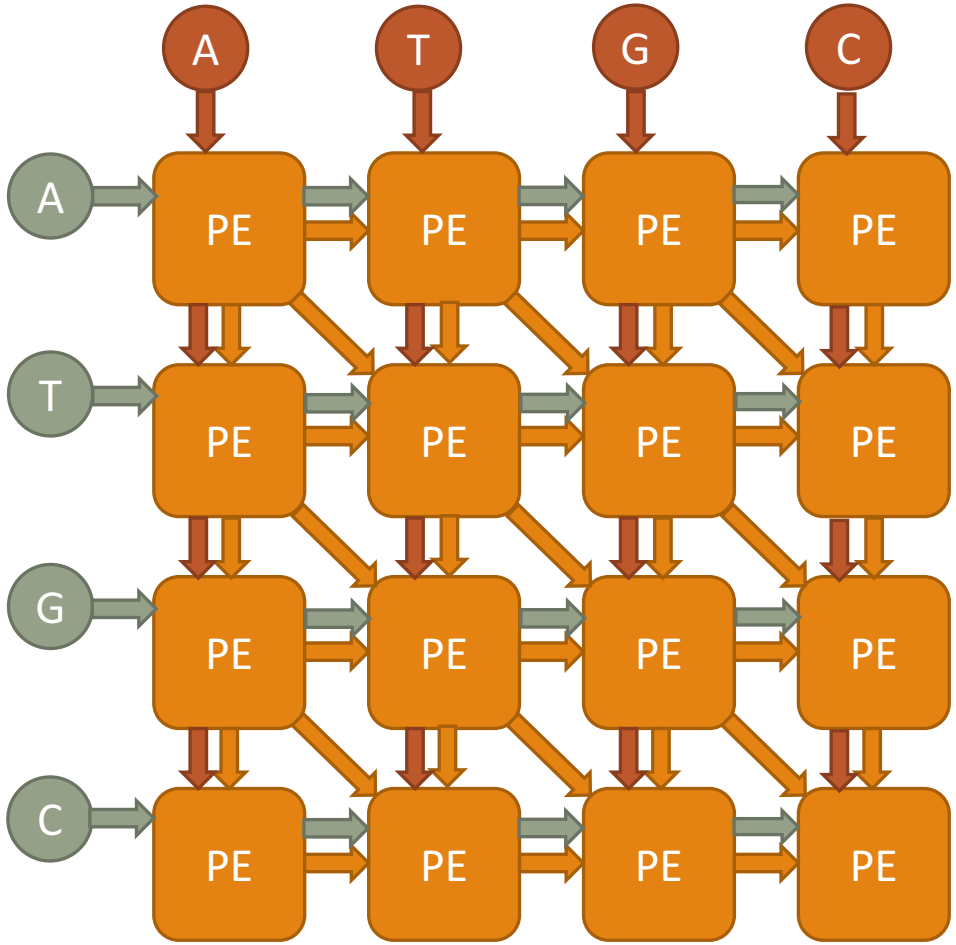
$$H(i,j) = \max \begin{Bmatrix} 0 & \\ H(i-1,j-1) + s(a_i,b_j) & \text{Match/Mismatch} \\ \max_{k \geq 1}\{H(i-k,j) + W_k\} & \text{Deletion} \\ \max_{l \geq 1}\{H(i,j-l) + W_l\} & \text{Insertion} \end{Bmatrix}, \ 1 \leq i \leq m, 1 \leq j \leq n$$

- $s(a,b) = +2$ if $a = b$ (match), -1 if $a \neq b$ (mismatch
- $W_i = -1$

$$H = \begin{pmatrix} & - & A & C & A & C & A & C & T & A \\ - & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ A & 0 & 2 & 1 & 2 & 1 & 2 & 1 & 0 & 2 \\ G & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ C & 0 & 0 & 3 & 2 & 3 & 2 & 3 & 2 & 1 \\ A & 0 & 2 & 2 & 5 & 4 & 5 & 4 & 3 & 4 \\ C & 0 & 1 & 4 & 4 & 7 & 6 & 7 & 6 & 5 \\ A & 0 & 2 & 3 & 6 & 6 & 9 & 8 & 7 & 8 \\ C & 0 & 1 & 4 & 5 & 8 & 8 & 11 & 10 & 9 \\ A & 0 & 2 & 3 & 6 & 7 & 10 & 10 & 10 & 12 \end{pmatrix}$$

$$T = \begin{pmatrix} & - & A & C & A & C & A & C & T & A \\ - & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ A & 0 & \nwarrow & \leftarrow & \nwarrow & \leftarrow & \nwarrow & \leftarrow & \leftarrow & \nwarrow \\ G & 0 & \uparrow & \nwarrow & \uparrow & \nwarrow & \uparrow & \nwarrow & \nwarrow & \uparrow \\ C & 0 & \uparrow & \nwarrow & \leftarrow & \nwarrow & \leftarrow & \nwarrow & \leftarrow & \leftarrow \\ A & 0 & \nwarrow & \uparrow & \nwarrow & \leftarrow & \nwarrow & \leftarrow & \leftarrow & \nwarrow \\ C & 0 & \uparrow & \nwarrow & \uparrow & \nwarrow & \leftarrow & \nwarrow & \leftarrow & \leftarrow \\ A & 0 & \nwarrow & \uparrow & \nwarrow & \uparrow & \nwarrow & \leftarrow & \leftarrow & \nwarrow \\ C & 0 & \uparrow & \nwarrow & \uparrow & \nwarrow & \uparrow & \nwarrow & \leftarrow & \leftarrow \\ A & 0 & \nwarrow & \uparrow & \nwarrow & \uparrow & \nwarrow & \uparrow & \nwarrow & \nwarrow \end{pmatrix}$$

Smith Waterman Implementation using Systolic Arrays



Smith Waterman Processing Element

- Compute current score from
  - Diagonal score
  - Top score
  - Left score
- Find maximum of the 3 scores to assign to the current element
- Combinational block executing every clock

- $s(a,b) = +2$ if $a = b$ (match), -1 if $a \neq b$ (mismatch
- $W_i = -1$

$$H = \begin{pmatrix} & - & A & C & A & C & A & C & T & A \\ - & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ A & 0 & 2 & 1 & 2 & 1 & 2 & 1 & 0 & 2 \\ G & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ C & 0 & 0 & 3 & 2 & 3 & 2 & 3 & 2 & 1 \\ A & 0 & 2 & 2 & 5 & 4 & 5 & 4 & 3 & 4 \\ C & 0 & 1 & 4 & 4 & 7 & 6 & 7 & 6 & 5 \\ A & 0 & 2 & 3 & 6 & 6 & 9 & 8 & 7 & 8 \\ C & 0 & 1 & 4 & 5 & 8 & 8 & 11 & 10 & 9 \\ A & 0 & 2 & 3 & 6 & 7 & 10 & 10 & 10 & 12 \end{pmatrix}$$

SW_PE_clk.v

```
18  // Additional Comments:
19  //
20  //////////////////////////////////////////////////////////////////////////////////
21  module SW_PE_clk(
22      input clk,
23      input reset,
24      input [2:0] seq1,
25      input [2:0] seq2,
26      input [31:0] diag_score,
27      input [31:0] left_score,
28      input [31:0] top_score,
29      output reg[31:0] score
30      );
31
32     parameter match_score = 2;
33     parameter mismatch_score = 1;
34     parameter gap_penalty = 1;
35
36     wire [31:0] score1,score2,score3,score_wire;
37     wire [31:0] stemp1,stemp2,stemp3;
38
39     assign stemp1 = ((seq1==seq2)?(diag_score+match_score):(diag_score-mismatch_score))&32'b10000000000000000000000000000000;
40     assign stemp2 = (left_score-gap_penalty)&32'b10000000000000000000000000000000;
41     assign stemp3 = (top_score-gap_penalty)&32'b10000000000000000000000000000000;
42
43     assign score1 = (stemp1==8'd0)?((seq1==seq2)?(diag_score+match_score):(diag_score-mismatch_score)):32'd0;
44     assign score2 = (stemp2==8'd0)?(left_score-gap_penalty):32'd0;
45     assign score3 = (stemp3==8'd0)?(top_score-gap_penalty):32'd0;
46
47     assign score_wire = ((score1>score2)?((score1>score3)?score1:score3):((score2>score3)?score2:score3));
48
49     always @(posedge clk or posedge reset) begin
50        if (reset) begin
51           score <= 32'd0;
52        end
53        else
54           score <= score_wire;
55     end
56
57  endmodule
58
```

Traceback Element

To output the character of sequence 1 and sequence 2 at (i,j)

- Diagonal value maximum : **seq1 and seq2 match** or **SNP**
- Left value maximum : **Deletion in seq2**
- Top value maximum : **Insertion in seq2**

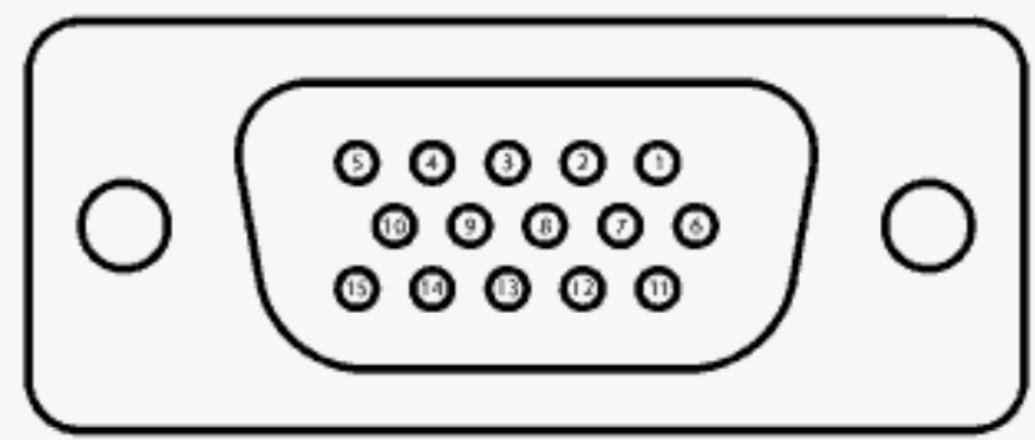
$$T = \begin{pmatrix} & - & A & C & A & C & A & C & T & A \\ - & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ A & 0 & \nwarrow & \leftarrow & \nwarrow & \leftarrow & \nwarrow & \leftarrow & \leftarrow & \nwarrow \\ G & 0 & \uparrow & \nwarrow & \uparrow & \nwarrow & \uparrow & \nwarrow & \nwarrow & \uparrow \\ C & 0 & \uparrow & \nwarrow & \leftarrow & \nwarrow & \leftarrow & \nwarrow & \leftarrow & \leftarrow \\ A & 0 & \nwarrow & \uparrow & \nwarrow & \leftarrow & \nwarrow & \leftarrow & \leftarrow & \nwarrow \\ C & 0 & \uparrow & \nwarrow & \uparrow & \nwarrow & \leftarrow & \nwarrow & \leftarrow & \leftarrow \\ A & 0 & \nwarrow & \uparrow & \nwarrow & \uparrow & \nwarrow & \leftarrow & \leftarrow & \nwarrow \\ C & 0 & \uparrow & \nwarrow & \uparrow & \nwarrow & \uparrow & \nwarrow & \leftarrow & \leftarrow \\ A & 0 & \nwarrow & \uparrow & \nwarrow & \uparrow & \nwarrow & \uparrow & \nwarrow & \nwarrow \end{pmatrix}$$

```
1  `timescale 1ns / 1ps
2  //////////////////////////////////////////////////////////////////////////////////
3  // Company: Boston University
4  // Engineer: Aastha Anand
5  //
6  // Create Date:    15:02:01 12/01/2016
7  // Design Name: Smith Waterman
8  // Module Name:    traceback_compare
9  // Project Name: EC551 Project
10 // Target Devices:
11 // Tool versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 //////////////////////////////////////////////////////////////////////////////////
21 module traceback_compare(curr_index,
22                           top_score,
23                           diag_score,
24                           left_score,
25                           current_score,
26                           //seq1,
27                          // seq2,
28                          // seq1_out,
29                          // seq2_out,
30                           next_index
31     );
32     parameter n =4;
33    input [31:0] top_score, diag_score, left_score, current_score;
34    input [n:0] curr_index;
35
36    output [n:0] next_index;
37
38    assign next_index = (diag_score<top_score)?((top_score>left_score)?curr_index-n-1:curr_index-1):((diag_score<left_score)?curr_index-1:curr_index-n-2)
39 endmodule
40
```

BOSTON UNIVERSITY

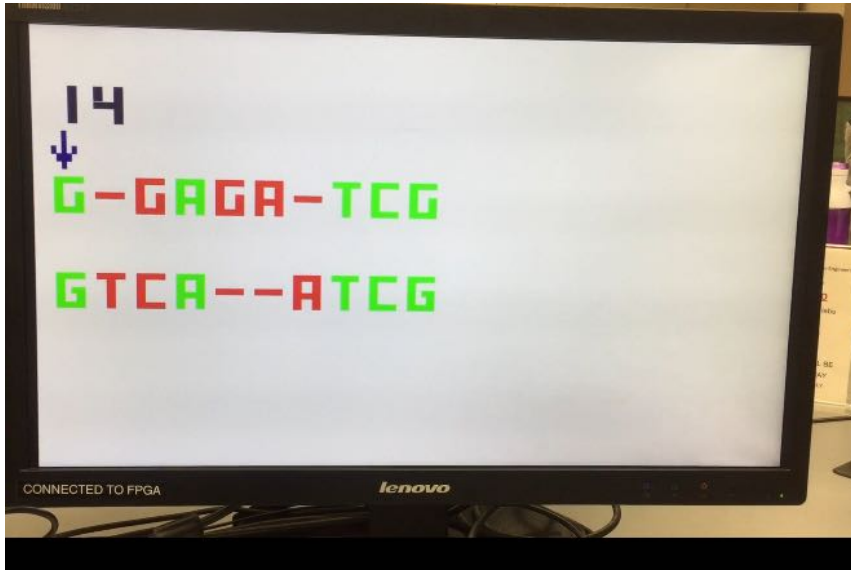
VGA Interfacing

- Zedboard allows 12 bit colour output through VGA connector
- So VGA_R, VGA_G, VGA_B all 4 bits each
- Other pins used are Horizantal sync and Vertical sync

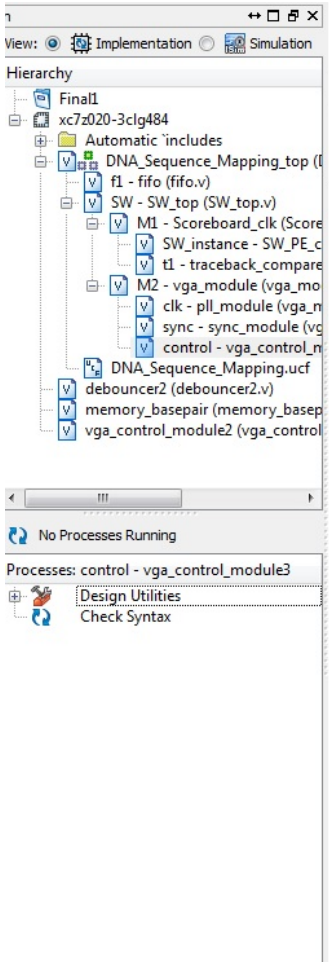


I/O Interfacing & challenges

- Serial Inputs from Smith Waterman block are displayed on the computer screen via VGA display cable
- Dynamically displaying all the sequences from the previous module along with the index as shown.
- Not easy because had to code for letters ‘A’, ‘G’, ‘C’, ‘T’ & ‘-’ .
- So coding for 15 characters in one sequence and 2 sequences in total!
- That’s 30 characters being displayed dynamically!
- Displaying the numbers dynamically by coding for digits ‘0 – 9’ for 2 digit index given by the Smith Waterman algorithm
- Too many if-else loop created too much constraints while synthesizing the code and it took nearly 30-40 mins for it to build.



Verilog Implementation of VGA Controller

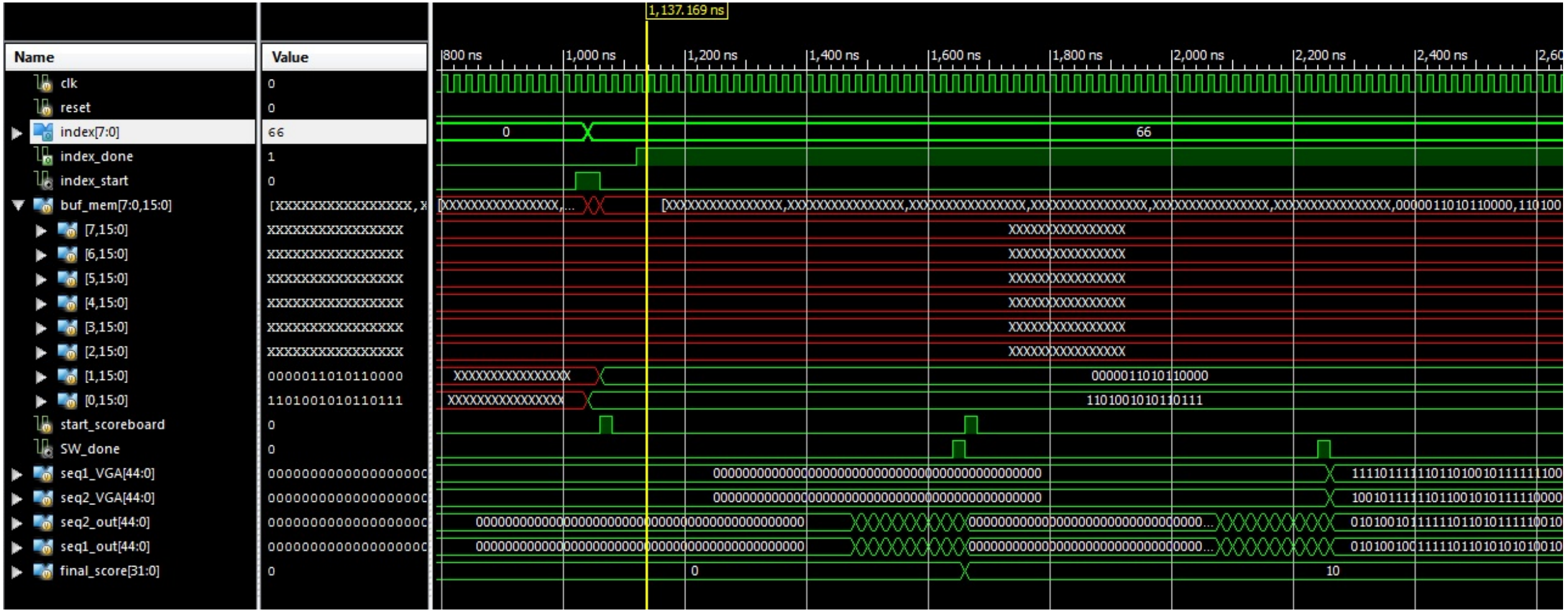



```
853             colouroutput[11:0]<= 12'b111100000000;
854         else if ((seq1[5:3]==3'b110) &&(((X>540 && X<570)&&(Y>210 && Y<220))||((X>540 && X<550)&&(Y>210 && Y<260))||((X>540 && X<570)&&(Y>250 && Y<260))))//"C"
855            if (seq2[5:3]==seq1[5:3])
856               colouroutput[11:0]<= 12'b000011110000;
857            else
858               colouroutput[11:0]<= 12'b111100000000;
859         else if ((seq1[5:3]==3'b111) &&(((X>540&&X<570)&&(Y>210&&Y<220))||((X>550&&X<560)&&(Y>210&&Y<260))))//"I"
860            if (seq2[5:3]==seq1[5:3])
861               colouroutput[11:0]<= 12'b000011110000;
862            else
863               colouroutput[11:0]<= 12'b111100000000;
864         else if ((seq1[5:3]==3'b010) && ((X>540&&X<570)&&(Y>230&&Y<240)))//"___"
865            colouroutput[11:0]<= 12'b111100000000;
866
867         else if ((seq1[2:0]==3'b100) &&(((X>580 && X<590)&&(Y>210 && Y<260))||((X>580 && X<610)&&(Y>210 && Y<220))||((X>600 && X<610)&&(Y>210 && Y<260))||((X>589 && X<601)&&(Y>235
868            if (seq2[2:0]==seq1[2:0])
869               colouroutput[11:0]<= 12'b000011110000;
870            else
871               colouroutput[11:0]<= 12'b111100000000;
872         else if ((seq1[2:0]==3'b101) &&(((X>580 && X<610)&&(Y>210 && Y<220))||((X>580 && X<590)&&(Y>210 && Y<260))||((X>580 && X<610)&&(Y>250 && Y<260))||((X>600 && X<610)&&(Y>230
873            if (seq2[2:0]==seq1[2:0])
874               colouroutput[11:0]<= 12'b000011110000;
875            else
876               colouroutput[11:0]<= 12'b111100000000;
877         else if ((seq1[2:0]==3'b110) &&(((X>580 && X<610)&&(Y>210 && Y<220))||((X>580 && X<590)&&(Y>210 && Y<260))||((X>580 && X<610)&&(Y>250 && Y<260))))//"C"
878            if (seq2[2:0]==seq1[2:0])
879               colouroutput[11:0]<= 12'b000011110000;
880            else
881               colouroutput[11:0]<= 12'b111100000000;
882         else if ((seq1[2:0]==3'b111) &&(((X>580&&X<610)&&(Y>210&&Y<220))||((X>590&&X<600)&&(Y>210&&Y<260))))//"I"
883            if (seq2[2:0]==seq1[2:0])
884               colouroutput[11:0]<= 12'b000011110000;
885            else
886               colouroutput[11:0]<= 12'b111100000000;
887         else if ((seq1[2:0]==3'b010) && ((X>580&&X<610)&&(Y>230&&Y<240)))//"___"
888            colouroutput[11:0]<= 12'b111100000000;
889
890         else
891            colouroutput[11:0]<= 12'b111111111111;
892      end
893   end
894   assign VGA_R = colouroutput[11:8];
895   assign VGA_G = colouroutput[7:4];
896   assign VGA_B = colouroutput[3:0];
897
898   endmodule
```

BOSTON UNIVERSITY

- Background and Motivation
- High Level Diagram and Specifications
- Functional Description of Blocks
- **Results and Analysis**
- Successes and Failures
- Conclusion and Future Work

BOSTON UNIVERSITY



Demo Movie 1 – Local Sequence Alignment

Seq1 – ACTGGTACTA
Seq2 – ACTGGACTGA

Objective – Find local sequence alignment of the 2 sequences

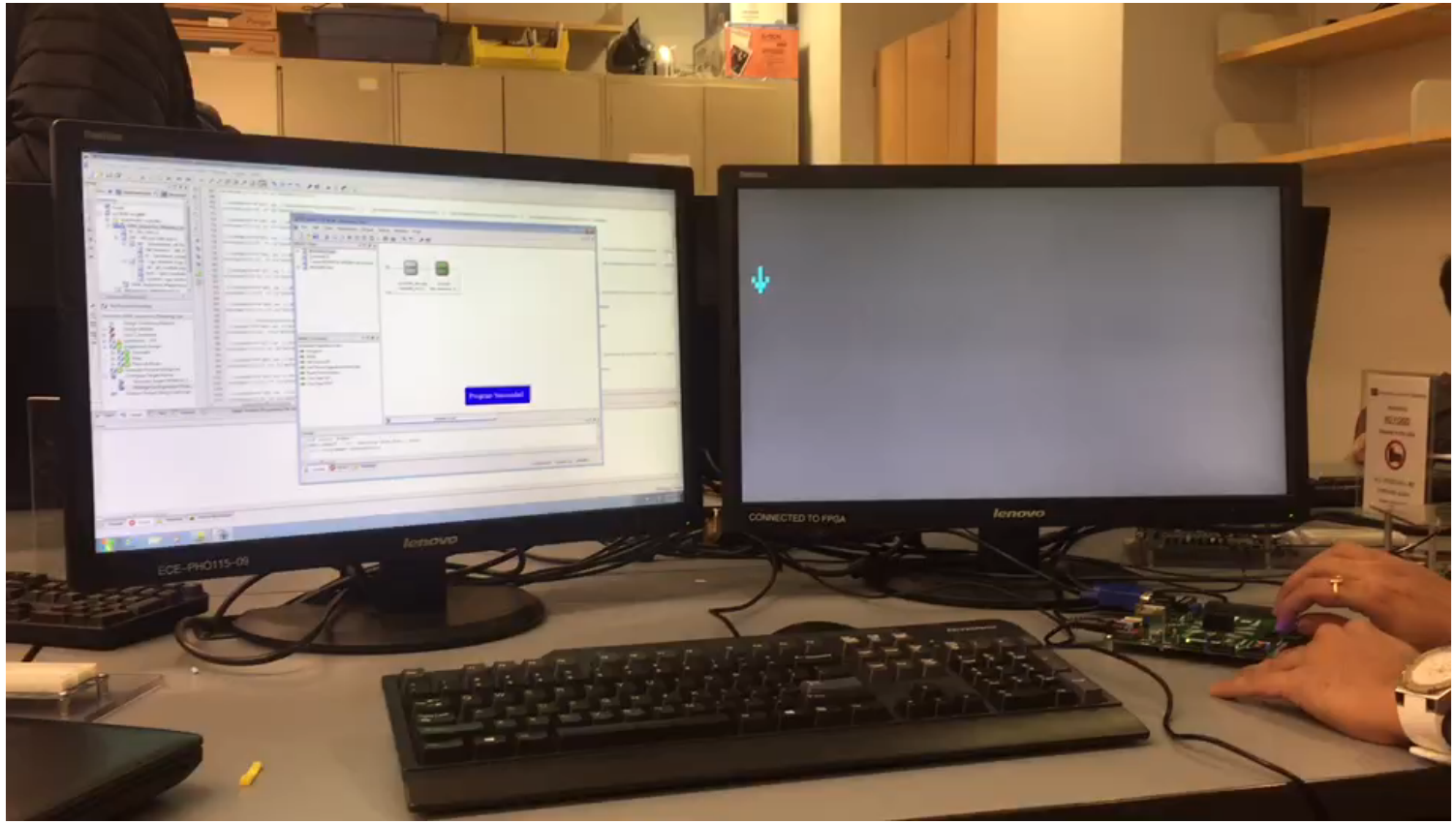
```
    input clk,reset;
    input start_scoreboard;
    output [`D_WIDTH-1:0] VGA_R, VGA_G, VGA_B;
   output VGA_HS, VGA_VS;
    wire [3*n-1:0] seq1,seq2;
    wire [2:0] A,T,C,G;
    assign A=3'b100;
    assign G=3'b101;
    assign C=3'b110;
    assign T=3'b111;
    //wire [44:0] seq1_out,seq2_out;
    reg [44:0] seq1_out,seq2_out;
      wire [2:0] seq1_out_SW,seq2_out_SW;
      wire out_valid;
    //assign seq1=45'b000001010111000000011010111011111000001011000;
   // assign seq2=45'b000001010111000000011010111011111000001011000;
   //assign seq1={A,C,A,C,G,C,T,A};
   //assign seq2={A,C,A,C,A,C,T,A};

   assign seq1={A,C,T,G,G,T,A,C,T,A};
   assign seq2={A,C,T,G,G,A,C,T,G,A};

   wire clock;
   reg start_SW;
      /*debouncer2 d1(.clock(clk),
                    .reset(reset),
                   .clock_new(clock)
                   );
```

Demo Movie 2

Ref Sequence: Length 50
TGTCCAGTATGCCTGACGTCGATCGCATTCGAGAATCCGAATAAAGTATT

Short Reads
1. TGGTAAAT
2. AGTCCGTA
3. CGTCGATC

Objective – Find the best match of the short reads in the reference sequence with substitution, insertions and deletions



BOSTON UNIVERSITY

- Background and Motivation
- High Level Diagram and Specifications
- Functional Description of Blocks
- Results and Analysis
- Successes and Failures
- Conclusion and Future Work

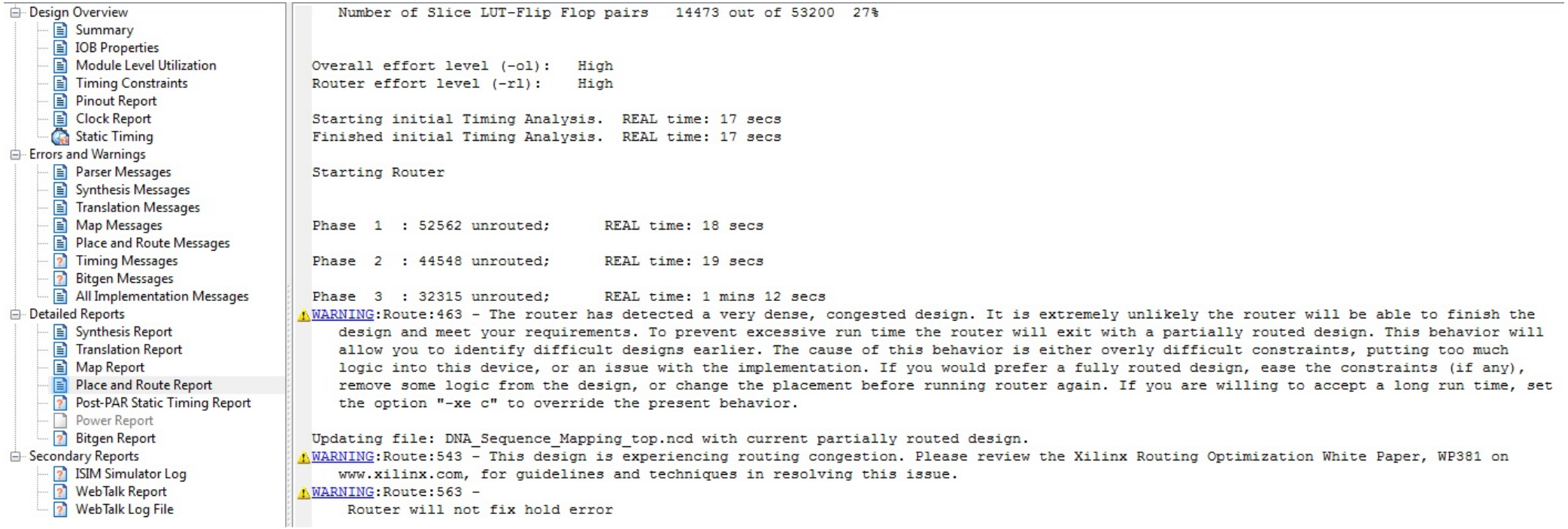
BOSTON UNIVERSITY

Successes

- ✓ Performed Local Alignment of 2 sequences using Smith Waterman Algorithm
- ✓ Implementation of CAL table for seeding the reference indexes
- ✓ VGA Interfacing for output display
- ✓ Timing Accurate Implementation of Smith Waterman Algorithm
- ✓ Working Prototype for performing DNA Sequence Mapping – Not that efficient

Failures

- Integration of Indexing and Smith Waterman Block -> Timing and Routing Error



Probably due to combinational implementation of Indexing

BOSTON UNIVERSITY

Failures

- Spent a lot of time on getting user input through UART (Putty)
- Tried Parallel Smith Waterman Implementation, but couldn't integrate with the top module
- Long Sequences takes a long time to build, (place & route) -> might need some compression algorithm
- Takes a long time for Place & Route (~45mins) -> Design not that efficient

BOSTON UNIVERSITY

Work Allocation

- High Level Block Diagram and Functional Specification – Everyone
- Indexing Block – Siva, Anand
- Smith Waterman Pre Processor – Anand
- Smith Waterman Block – Aditya, Aastha
- VGA Interfacing for I/O – Sarthak, Aastha, Aditya
- Overall Integration – Aditya, Siva, Sarthak

BOSTON UNIVERSITY

- Background and Motivation
- High Level Diagram and Specifications
- Functional Description of Blocks
- Results and Analysis
- Successes and Failures
- Conclusion and Future Work

BOSTON UNIVERSITY

Future Work

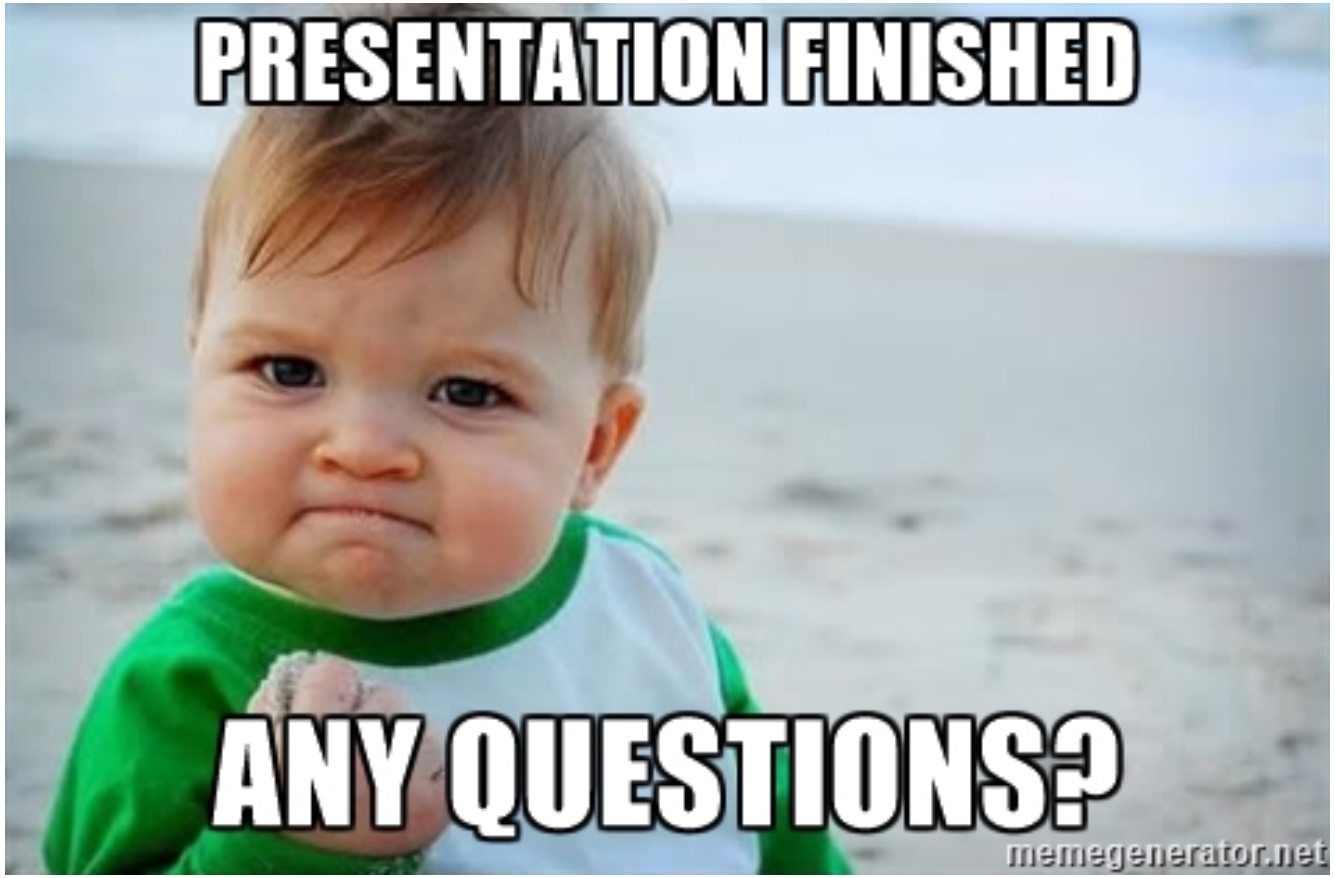
- Parallel Smith Waterman algorithm
- Test with longer short reads and references
- Store Reference sequence in DRAM and implement a DRAM controller
- Burrows-Wheeler Transform for data compression to solve the short read mapping problem

BOSTON UNIVERSITY

References

- Olson, Corey B., et al. "Hardware acceleration of short read mapping." *Field-Programmable Custom Computing Machines (FCCM), 2012 IEEE 20th Annual International Symposium on*. IEEE, 2012.
- Lopresti, Daniel P. "P-NAC: A systolic array for comparing nucleic acid sequences." *Computer* 20.7 (1987): 98-99.
- Herbordt, Martin C., et al. "Single pass streaming BLAST on FPGAs." *Parallel computing* 33.10 (2007): 741-756.
- https://en.wikipedia.org/wiki/Smith%E2%80%93Waterman_algorithm
- Vermij, E. P. *Genetic sequence alignment on a supercomputing platform*. Diss. TU Delft, Delft University of Technology, 2011.

- Special Thanks to Prof. Martin Herbordt and Prof. Wenchao Li for inputs


PRESENTATION FINISHED
ANY QUESTIONS?
memegenerator.net