

# FPGA Acceleration of DNA Sequence Mapping

Team: DNA Mappers

---

ADITYA NARAYAN

AASTHA ANAND

G SIVA PERUMAL

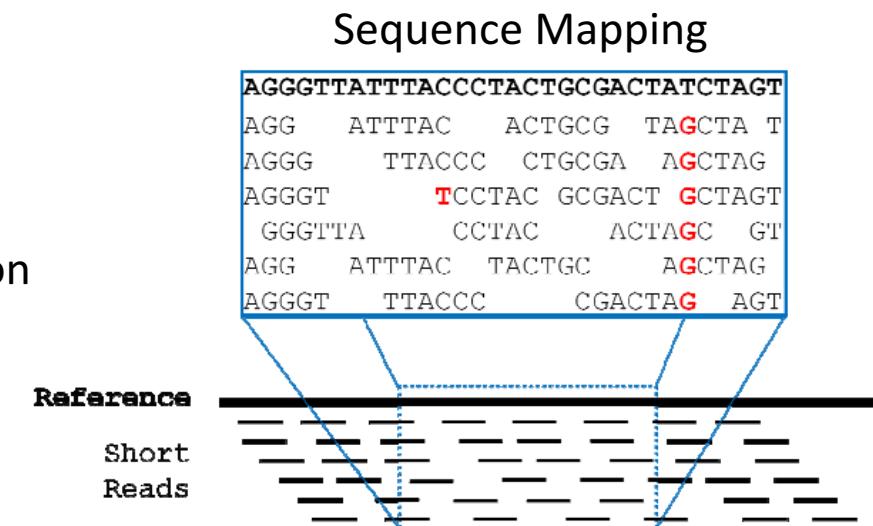
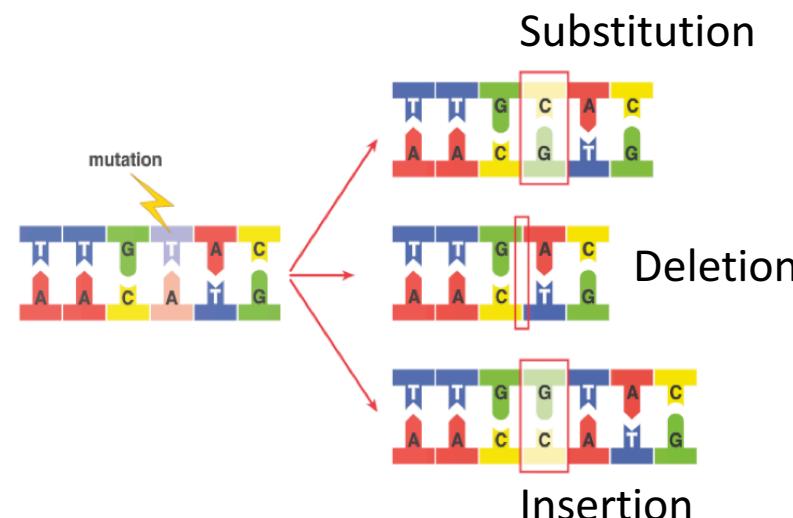
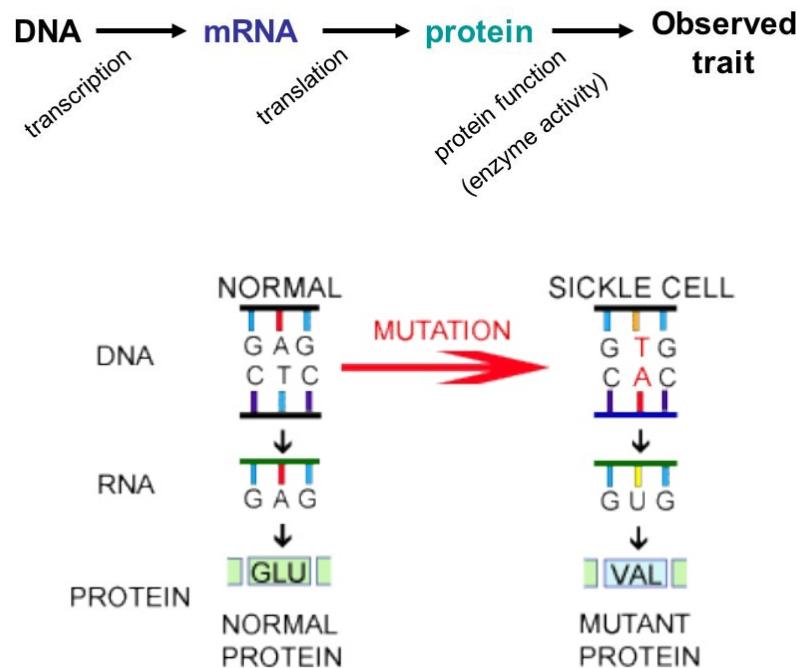
SARTHAK JAGETIA

ANAND SHIVALKAR

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

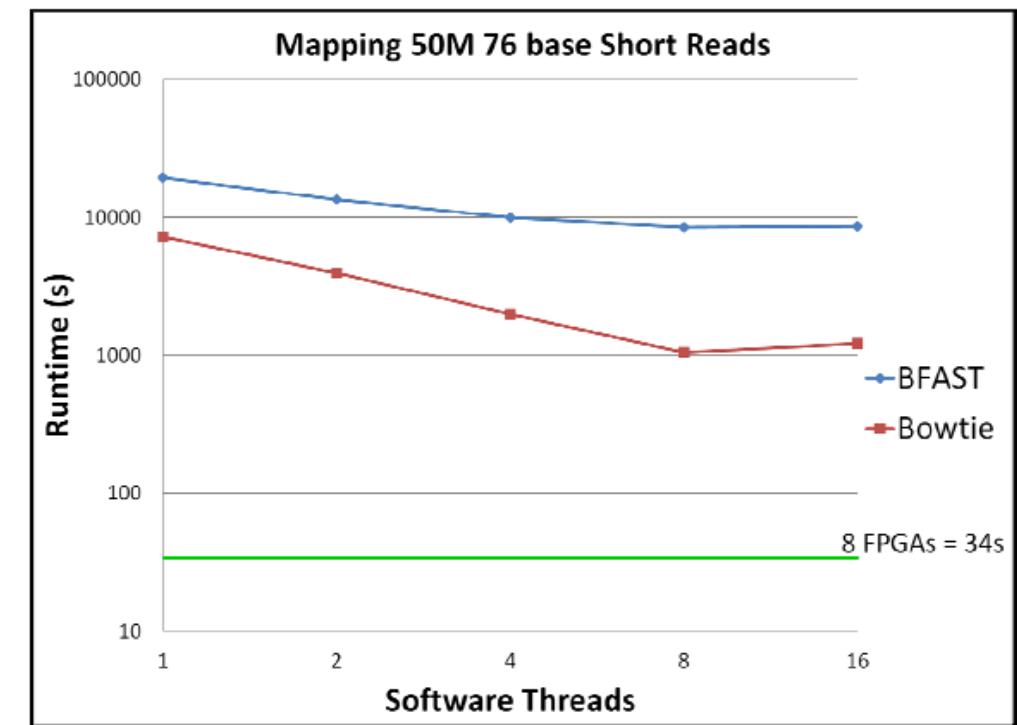
# Background

## 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



The screenshot shows the DRAGEN Bio-IT Processor website. At the top, there's a navigation bar with links for 'Get DRAGEN Now', 'Customer Login', and social media icons. Below the navigation is a large banner featuring a white dragon logo and the text 'DRAGEN™ Bio-IT Processor'. A central image shows a black DRAGEN processor chip with a label that reads 'DRAGEN by edico genome DRGN5500-001 422SEXSQLJCA V0.4-000'. Below the banner, a dark orange background with a wavy pattern contains the text 'the world's first NGS bioinformatics processor'.

## DRAGEN Bio-IT Platform

Enabling the Global Genomic Infrastructure

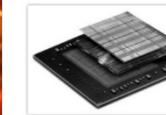
The DRAGEN (Dynamic Read Analysis for Genomics) Platform is based on the highly reconfigurable DRAGEN Bio-IT Processor. The DRAGEN Processor uses a field-programmable gate array (FPGA) to provide hardware-accelerated implementations of genome pipeline algorithms, such as BCL conversion, compression, 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.

## THE NEXT PLATFORM

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.

## THE NEXT PLATFORM

HOME COMPUTE STORE CONNECT CONTROL CODE ANALYZE HPC ENTERPRISE

### GOOGLE, CYCLE COMPUTING PAIR FOR BROAD GENOMICS EFFORT

September 8, 2015 Nicole Hemsoth

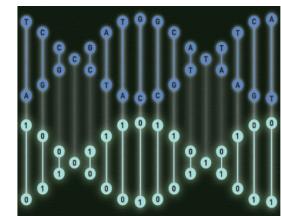


In times long since passed, when cloud was the latest, greatest hype machine, a steady wave of "cloud enablement" companies came to fore, promising secure ease as users tiptoed across the firewall. As we have seen over the last several years, a great many of these have been absorbed by infrastructure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

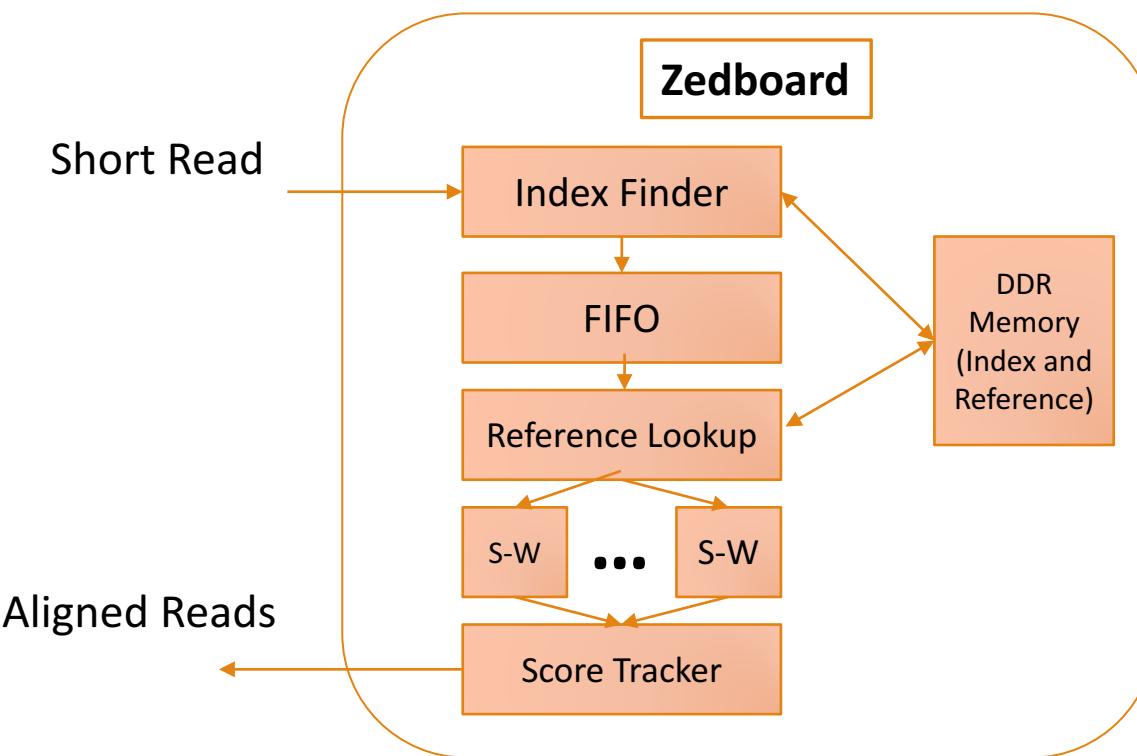
Cambridge, MA, November 17th, 2016

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.

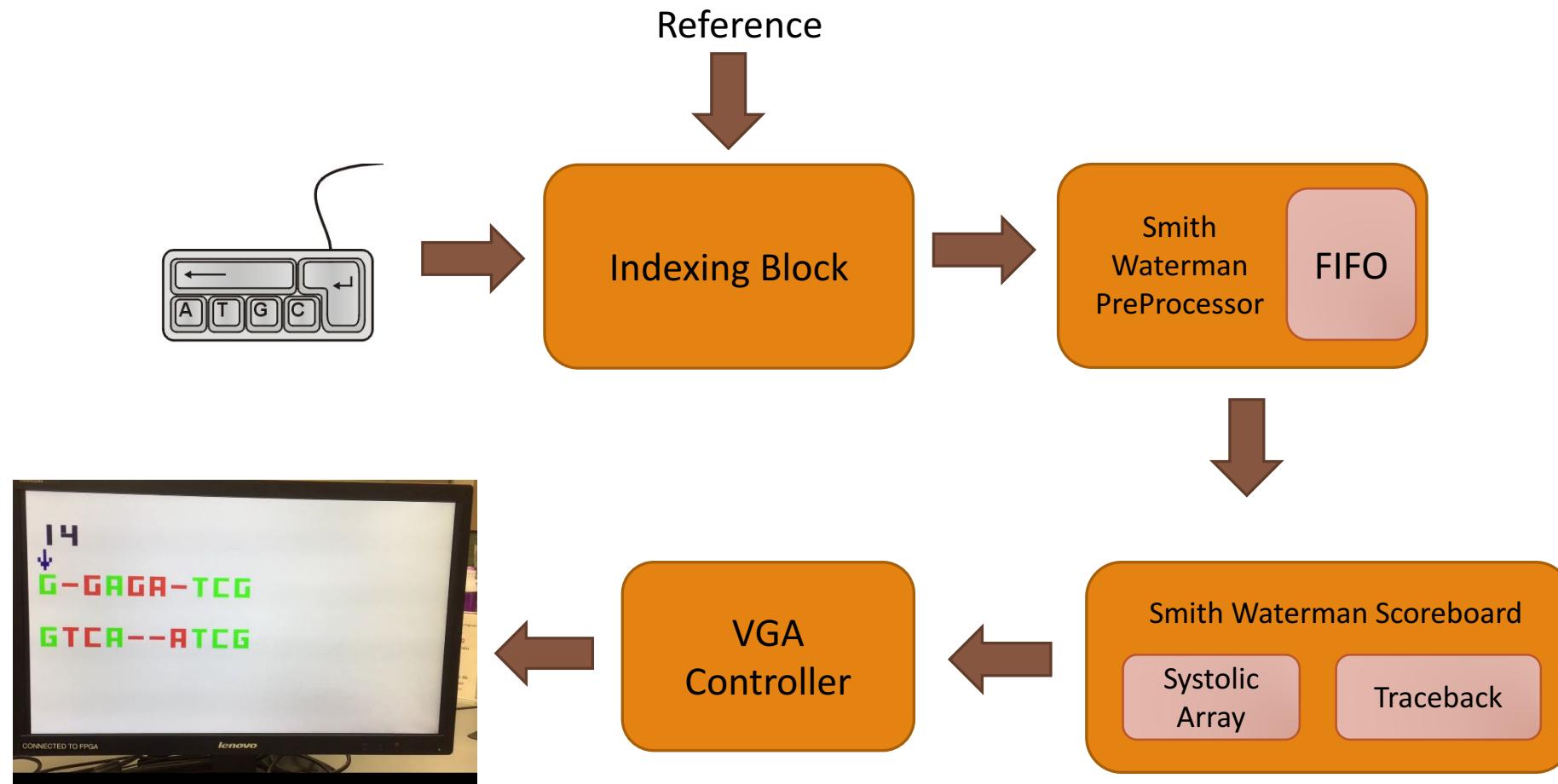
Through a five-year collaboration, researchers and software engineers at the new Intel-Broad Center for Genomic Data Engineering will build, optimize, and widely share new tools and infrastructure that will help scientists integrate and process genomic data. The project aims to optimize best practices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

- 
- **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



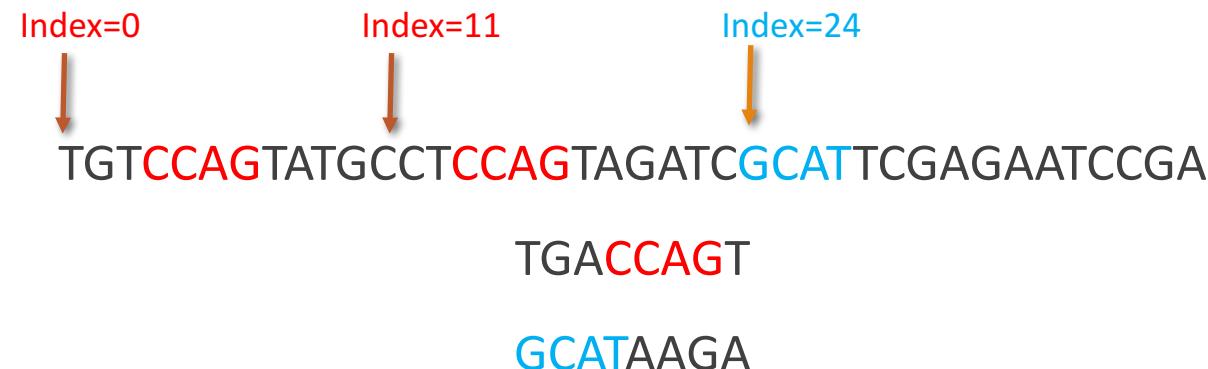
# Final High Level Block Diagram



- 
- **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 ( 1 = 0;1 <61; 1 = 1+1)
begin
tempreg_a = mem_a[1];
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];
```

# 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

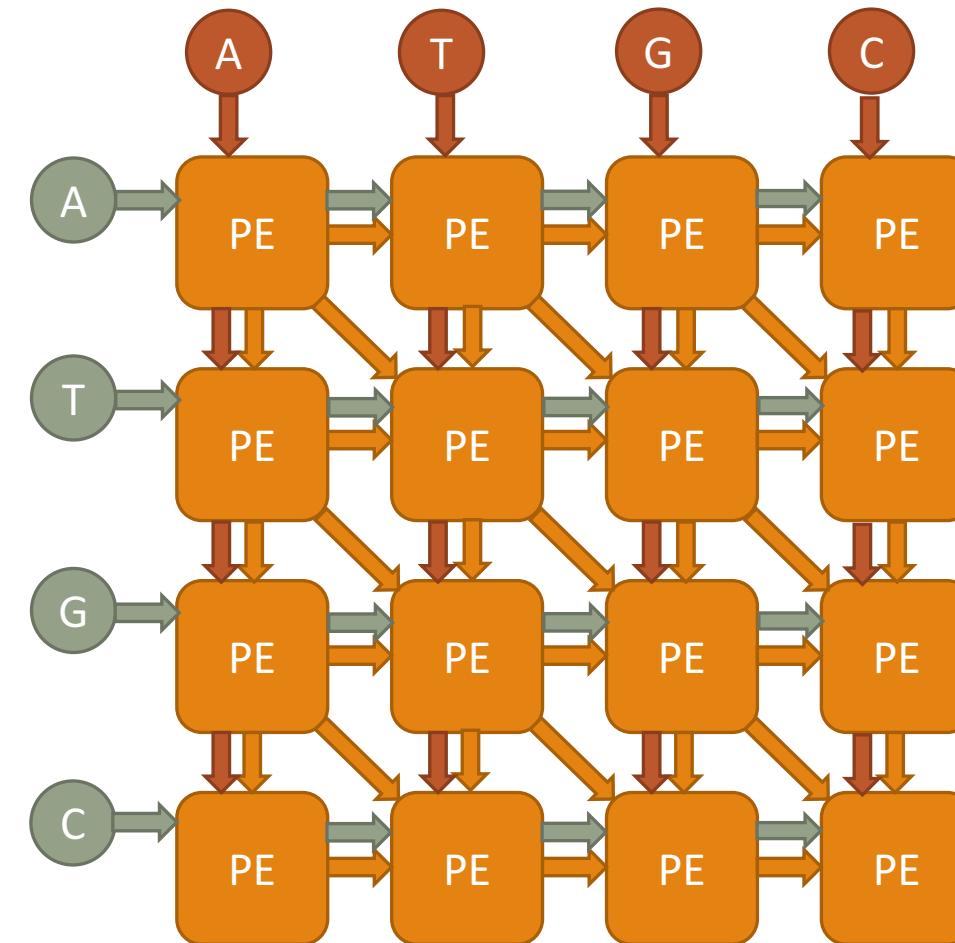
$$H(i, j) = \max \left\{ \begin{array}{l} H(i - 1, j - 1) + s(a_i, b_j) \\ \max_{k \geq 1} \{H(i - k, j) + W_k\} \\ \max_{l \geq 1} \{H(i, j - l) + W_l\} \end{array} \right. \begin{array}{l} \text{Match/Mismatch} \\ \text{Deletion} \\ \text{Insertion} \end{array} \right\}, \quad 1 \leq i \leq m, 1 \leq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

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

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

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

# 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'b100000000000000000000000000000000000000000000000000000000000000;
40 assign stemp2 = (left_score-gap_penalty)&32'b100000000000000000000000000000000000000000000000000000000000000;
41 assign stemp3 = (top_score-gap_penalty)&32'b100000000000000000000000000000000000000000000000000000000000000;
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**

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

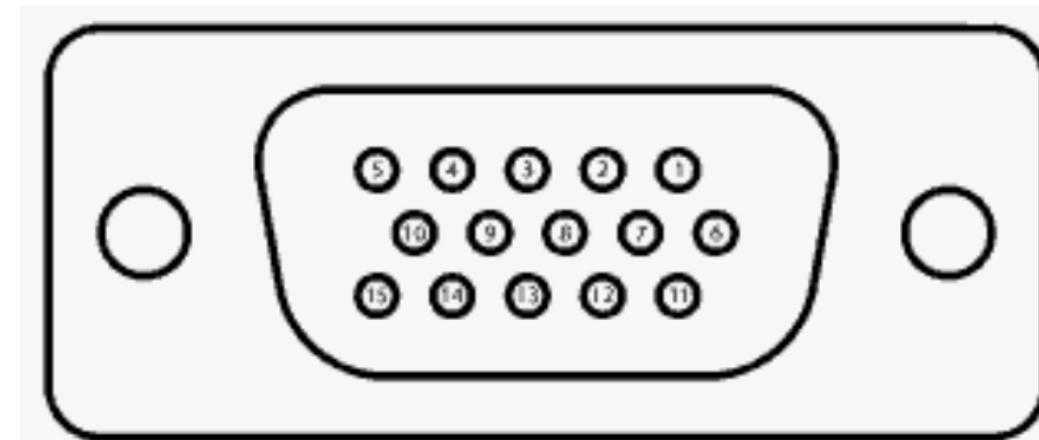
```

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

```

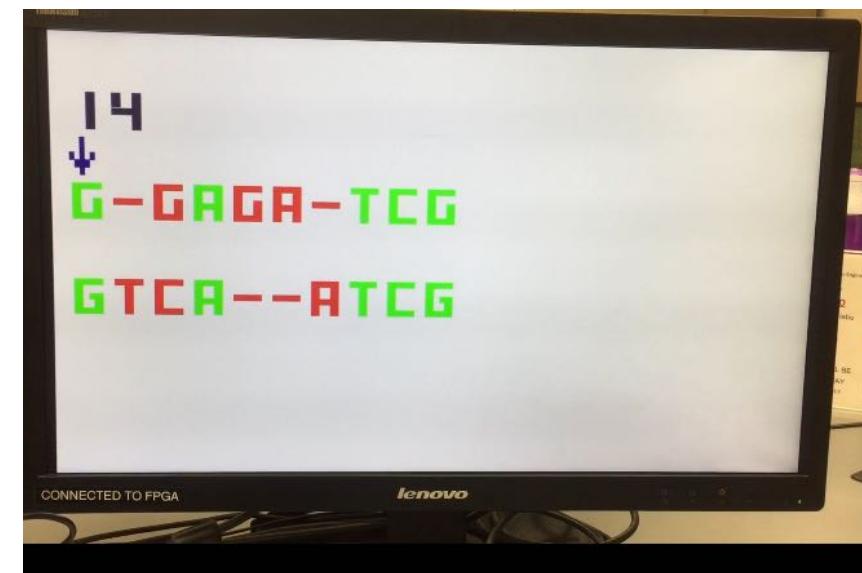
# VGA Interfacing

- Zedboard allows 12 bit colour output through VGA connector
- So VGA\_R, VGA\_G, VGA\_B all 4 bits each
- Other pins used are Horizontal 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.



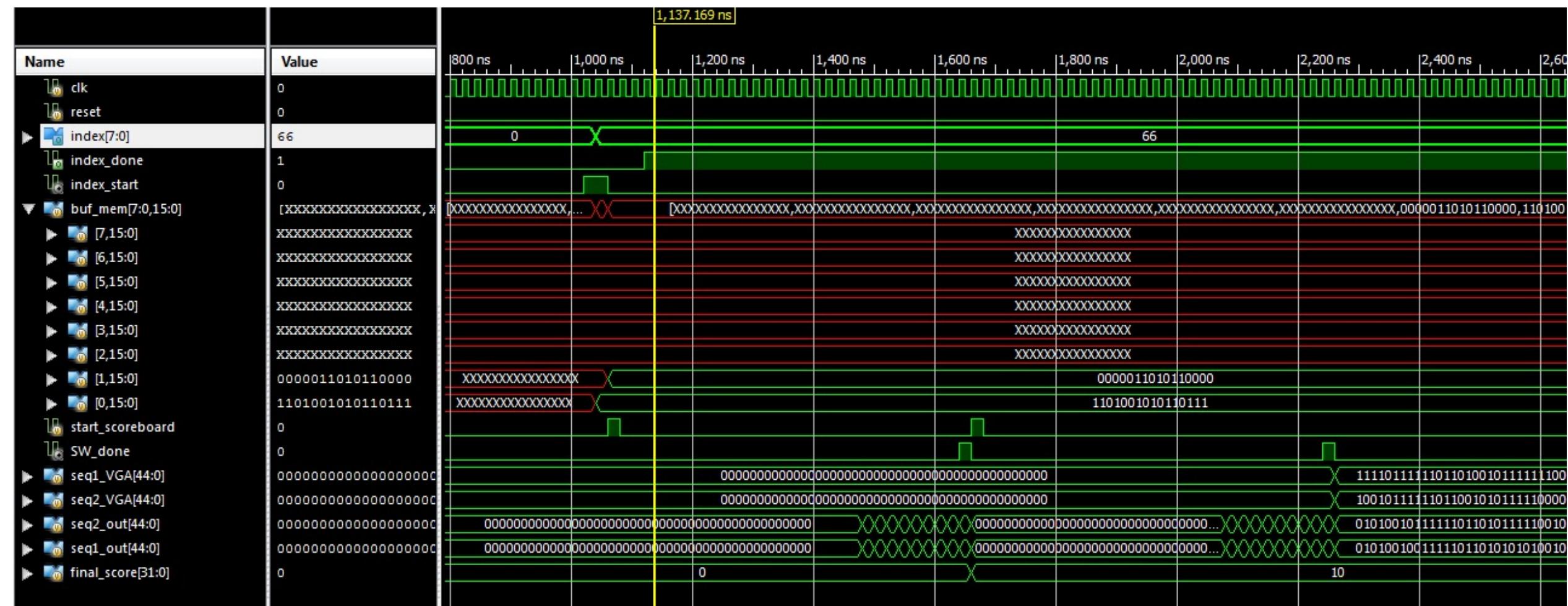
# Verilog Implementation of VGA Controller

The screenshot shows a Verilog IDE interface with the following details:

- Hierarchy:** The project structure is displayed on the left, showing modules like `Final1`, `xc7z020-3clg484`, `Automatic includes`, `DNA_Sequence_Mapping_top`, `f1 - fifo (fifo.v)`, `SW - SW_top (SW_top.v)`, `M1 - Scoreboard_clk (Scoreboard.v)`, `SW_instance - SW_PE_C (SW_instance.v)`, `tl - traceback_compare (traceback_compare.v)`, `M2 - vga_module (vga_module.v)`, `clk - pll_module (vga_pll.v)`, `sync - sync_module (sync.v)`, `control - vga_control (vga_control.v)`, `DNA_Sequence_Mapping.ucf`, `debouncer2 (debouncer2.v)`, `memory_basepair (memory_basepair.v)`, and `vga_control_module2 (vga_control.v)`.
- Code View:** The main window displays the Verilog code for the `vga_control_module2` module. The code handles the generation of VGA output colors based on sequence numbers and coordinates (`X` and `Y`). It includes logic for generating colors for specific patterns like "C" and "T". The code spans lines 853 to 898.

```
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))) // "T"
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))) // "T"
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
```

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



# Demo Movie 1 – Local Sequence Alignment

**Seq1** – ACTGGTACTA

**Seq2** – ACTGGACTGA

**Objective** – Find local sequence alignment of the 2 sequences

```
52      input clk,reset;
53      input start_scoreboard;
54      output [`D_WIDTH-1:0] VGA_R, VGA_G, VGA_B;
55      output VGA_HS, VGA_VS;
56      wire [3*n-1:0] seq1,seq2;
57      wire [2:0] A,T,C,G;
58      assign A=3'b100;
59      assign G=3'b101;
60      assign C=3'b110;
61      assign T=3'b111;
62      //wire [44:0] seq1_out,seq2_out;
63      reg [44:0] seq1_out,seq2_out;
64      wire [2:0] seq1_out_SW,seq2_out_SW;
65      wire out_valid;
66      //assign seq1=45'b00000101011100000001101011101111000001011000;
67      // assign seq2=45'b00000101011100000001101011101111000001011000;
68      //assign seq1={A,C,A,C,G,C,T,A};
69      //assign seq2={A,C,A,C,A,C,T,A};
70
71      assign seq1={A,C,T,G,G,T,A,C,T,A};
72      assign seq2={A,C,T,G,G,A,C,T,G,A};
73
74      wire clock;
75      reg start_SW;
76      /*debouncer2 d1(.clock(clk),
77                  .reset(reset),
78                  .clock_new(clock)
79 );
```

# Demo Movie 2

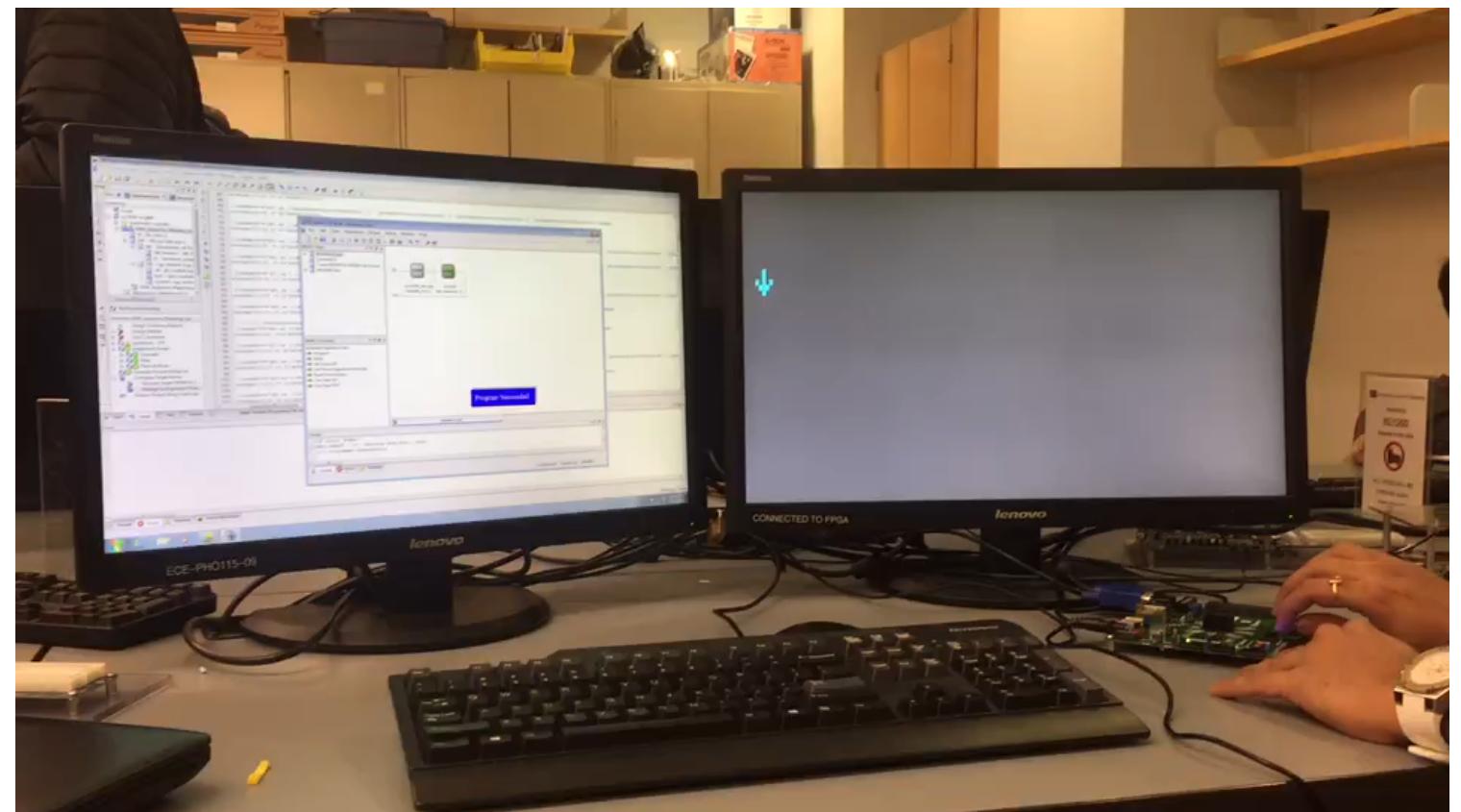
**Ref Sequence:** Length 50

TGTCCAGTATGCCTGACGTCGATCGCATTGAGAATCGAATAAAAGTATT

**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



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

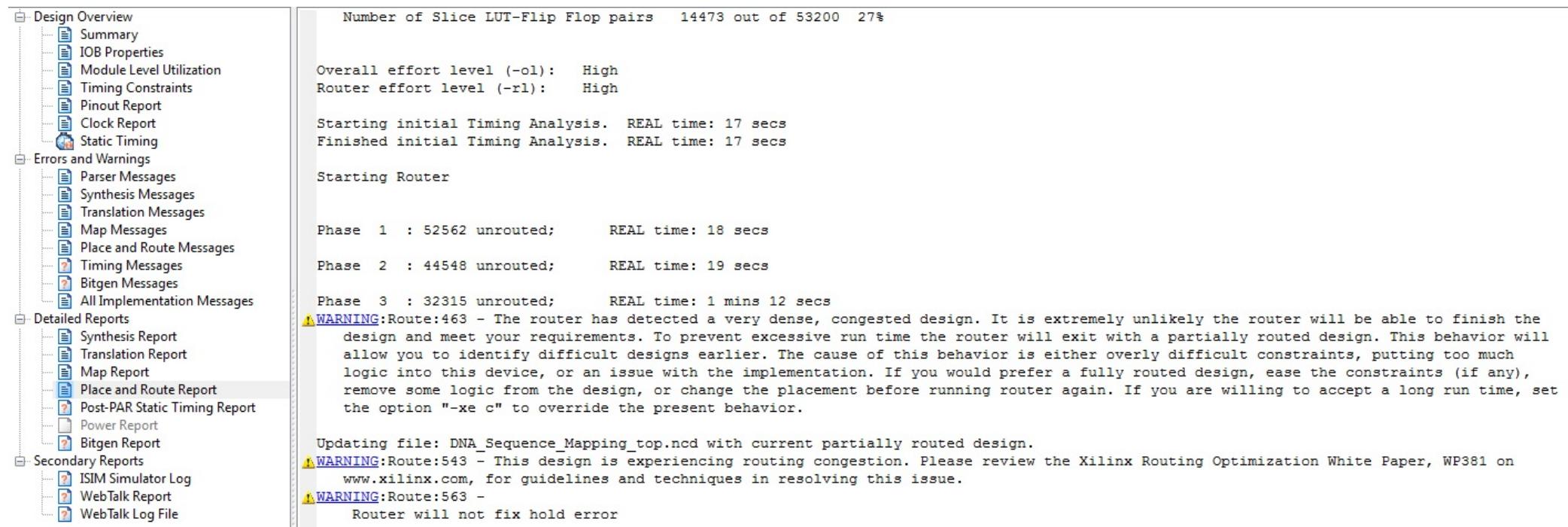
# Successes

---

- ✓ 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



The screenshot shows a software interface with a navigation tree on the left and a log output on the right.

**Navigation Tree:**

- Design Overview
  - Summary
  - IOB Properties
  - Module Level Utilization
  - Timing Constraints
  - Pinout Report
  - Clock Report
  - Static Timing
- Errors and Warnings
  - Parser Messages
  - Synthesis Messages
  - Translation Messages
  - Map Messages
  - Place and Route Messages
  - Timing Messages
  - Bitgen Messages
  - All Implementation Messages
- Detailed Reports
  - Synthesis Report
  - Translation Report
  - Map Report
  - Place and Route Report
  - Post-PAR Static Timing Report
  - Power Report
  - Bitgen Report
- Secondary Reports
  - ISIM Simulator Log
  - WebTalk Report
  - WebTalk Log File

**Log Output (Right Panel):**

```
Number of Slice LUT-Flip Flop pairs 14473 out of 53200 27%
```

Overall effort level (-ol): High  
Router effort level (-rl): High

Starting initial Timing Analysis. REAL time: 17 secs  
Finished initial Timing Analysis. REAL time: 17 secs

Starting Router

Phase 1 : 52562 unrouted; REAL time: 18 secs

Phase 2 : 44548 unrouted; REAL time: 19 secs

Phase 3 : 32315 unrouted; REAL time: 1 mins 12 secs

**WARNING:Route:463** - The router has detected a very dense, congested design. It is extremely unlikely the router will be able to finish the design and meet your requirements. To prevent excessive run time the router will exit with a partially routed design. This behavior will allow you to identify difficult designs earlier. The cause of this behavior is either overly difficult constraints, putting too much logic into this device, or an issue with the implementation. If you would prefer a fully routed design, ease the constraints (if any), remove some logic from the design, or change the placement before running router again. If you are willing to accept a long run time, set the option "-xe c" to override the present behavior.

Updating file: DNA\_Sequence\_Mapping\_top.ncd with current partially routed design.

**WARNING:Route:543** - This design is experiencing routing congestion. Please review the Xilinx Routing Optimization White Paper, WP381 on www.xilinx.com, for guidelines and techniques in resolving this issue.

**WARNING:Route:563** - Router will not fix hold error

Probably due to combinational implementation of Indexing

# 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

# 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**

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

# 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

# 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](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

