FPGA Report

Implementation of FHAST on

Vitis HLS

Team members:

V.Karthikeya(IMT2021504)

Rohit Mogli(IMT2021503)

NVS Asrith(IMT2021508)

Introduction:

Traditional DNA sequencing tools like BOWTIE often have slow processing speeds due to computational bottlenecks. To solve this, FHAST, an FPGA-based replacement, was introduced. It speeds up the process by using parallelism and high memory bandwidth, achieving up to 70x faster performance compared to single-threaded BOWTIE and 12x faster than eight-threaded BOWTIE, while keeping the mapping accuracy nearly the same.

FHAST uses the FM-Index to quickly search for patterns in the reference genome. It processes reads using multiple hardware threads to reduce memory delays and improve speed. The system also uses precomputed memory addresses and external memory to handle large genomes, making FHAST much faster than traditional tools for DNA sequencing.

Some of the processes involved in performing FM Index are

Burrows-Wheeler Transform (BWT): BWT rearranges a sequence to group similar patterns, making it efficient for locating and matching sequences.

Suffix Array: Suffix Array is a sorted list of all possible suffixes of a sequence, which allows quick access to where specific patterns occur in the sequence.

C-Table: C-Table is a table that counts the occurrences of each character in the sequence up to a certain point helping to quickly narrow down the search range when looking up patterns.

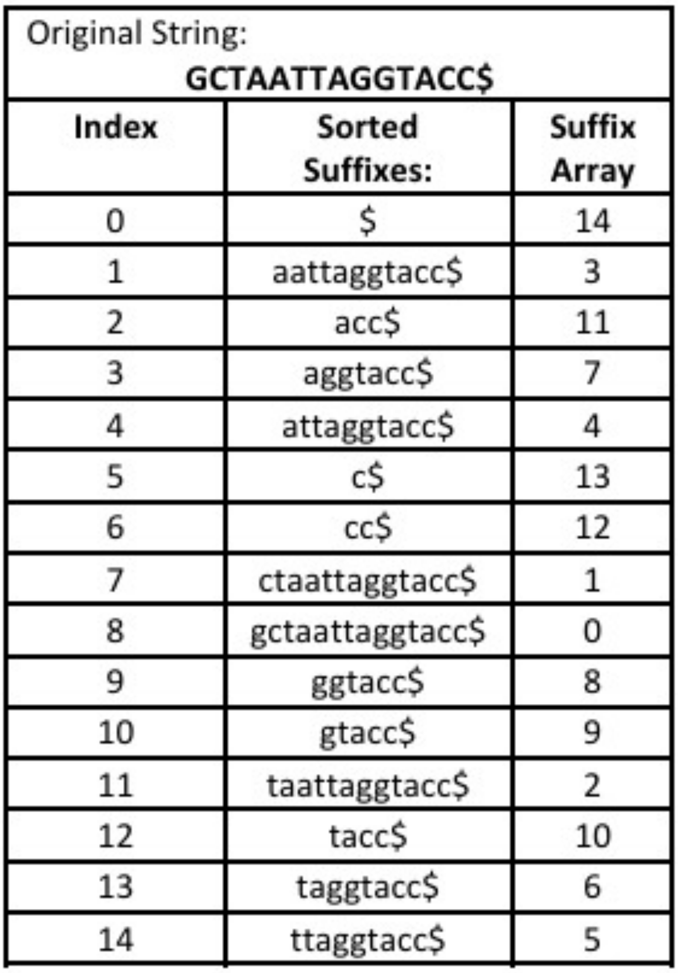
I-Table: The I-Table stores the first occurrence of the each character in SBWT(first element of each string in suffix array). This would be helpful for the calculating the occurrence of the pattern.

Searching: When you search for a pattern, the FM-Index uses two markers (called "top" and "bottom") to narrow down where the pattern might be in the text. As it goes through the characters in the pattern, it keeps updating these markers to zoom in on the possible positions.

$$bottom_{new} = C - table[n, Bottom_{current} + I - table[n]$$

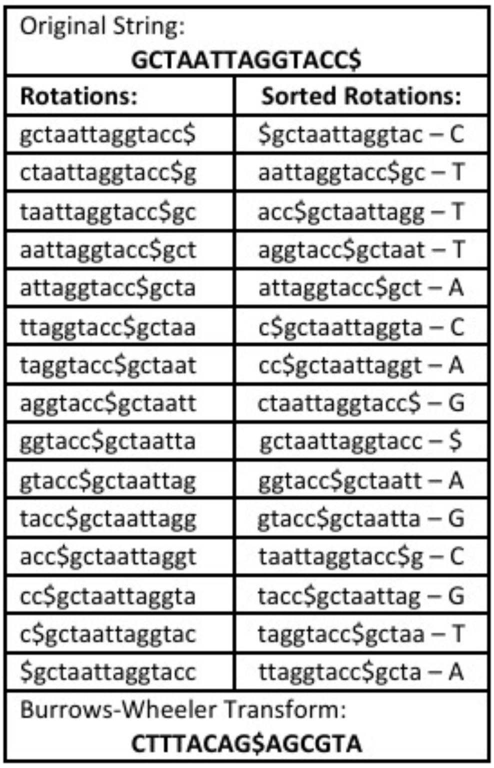
$$top_{new} = C - table[n, Top_{current} + I - table[n]$$

Example : GCTAATTAGGTACC$



Original String:
GCTAATTAGGTACC$

Index	**Sorted Suffixes:**	**Suffix Array**
0	$	14
1	aattaggtacc$	3
2	acc$	11
3	aggtacc$	7
4	attaggtacc$	4
5	c$	13
6	cc$	12
7	ctaattaggtacc$	1
8	gctaattaggtacc$	0
9	ggtacc$	8
10	gtacc$	9
11	taattaggtacc$	2
12	tacc$	10
13	taggtacc$	6
14	ttaggtacc$	5

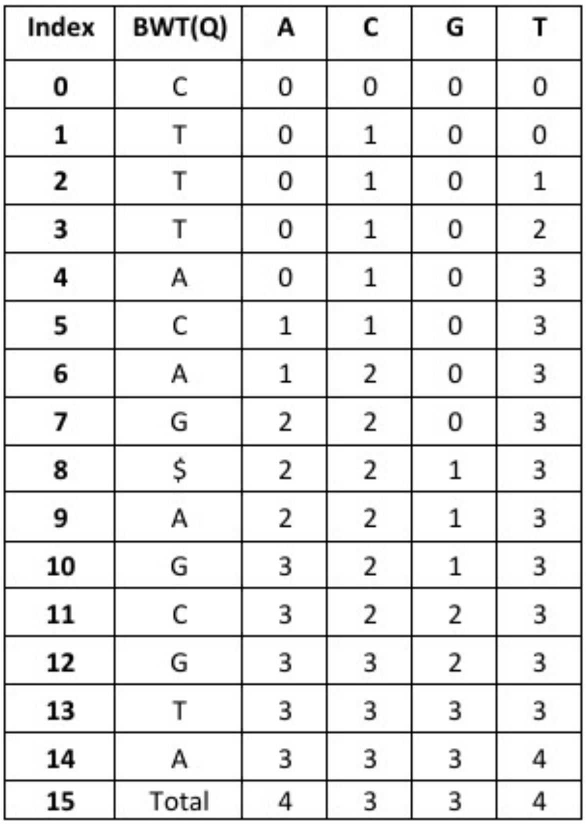


Original String:
GCTAATTAGGTACC$

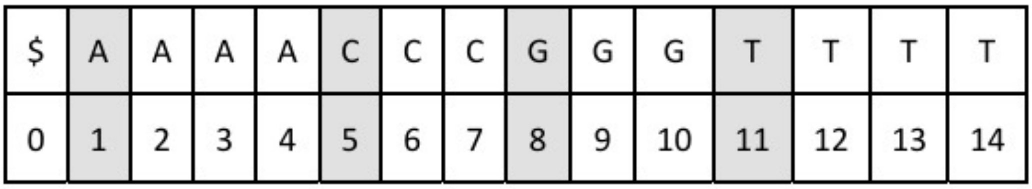
Rotations:	**Sorted Rotations:**
gctaattaggtacc$	$gctaattaggtac – C
ctaattaggtacc$g	aattaggtacc$gc – T
taattaggtacc$gc	acc$gctaattagg – T
aattaggtacc$gct	aggtacc$gctaat – T
attaggtacc$gcta	attaggtacc$gct – A
ttaggtacc$gctaa	c$gctaattaggta – C
taggtacc$gctaat	cc$gctaattaggt – A
aggtacc$gctaatt	ctaattaggtacc$ – G
ggtacc$gctaatta	gctaattaggtacc – $
gtacc$gctaattag	ggtacc$gctaatt – A
tacc$gctaattagg	gtacc$gctaatta – G
acc$gctaattaggt	taattaggtacc$g – C
cc$gctaattaggta	tacc$gctaattag – G
c$gctaattaggtac	taggtacc$gctaa – T
$gctaattaggtacc	ttaggtacc$gcta – A

Burrows-Wheeler Transform:
CTTTACAG$AGCGTA

C-table



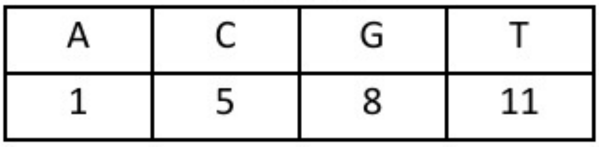
Index	BWT(Q)	A	C	G	T
0	C	0	0	0	0
1	T	0	1	0	0
2	T	0	1	0	1
3	T	0	1	0	2
4	A	0	1	0	3
5	C	1	1	0	3
6	A	1	2	0	3
7	G	2	2	0	3
8	$	2	2	1	3
9	A	2	2	1	3
10	G	3	2	1	3
11	C	3	2	2	3
12	G	3	3	2	3
13	T	3	3	3	3
14	A	3	3	3	4
15	Total	4	3	3	4



$	A	A	A	A	C	C	C	G	G	G	T	T	T	T
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

SBWT(Q) for Q = GCTAATTAGGTACC$

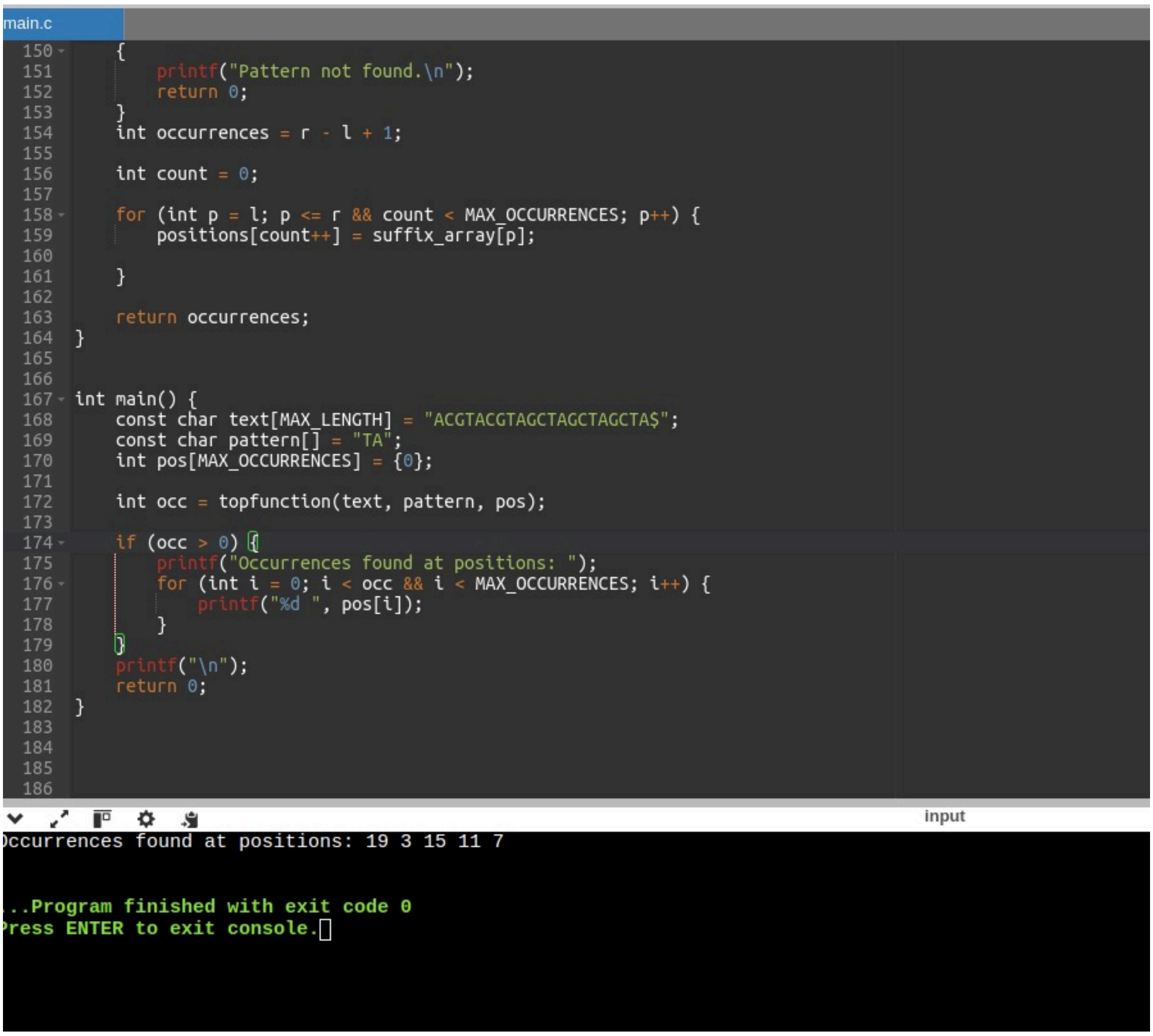
I-table



A	C	G	T
1	5	8	11

Methodology:

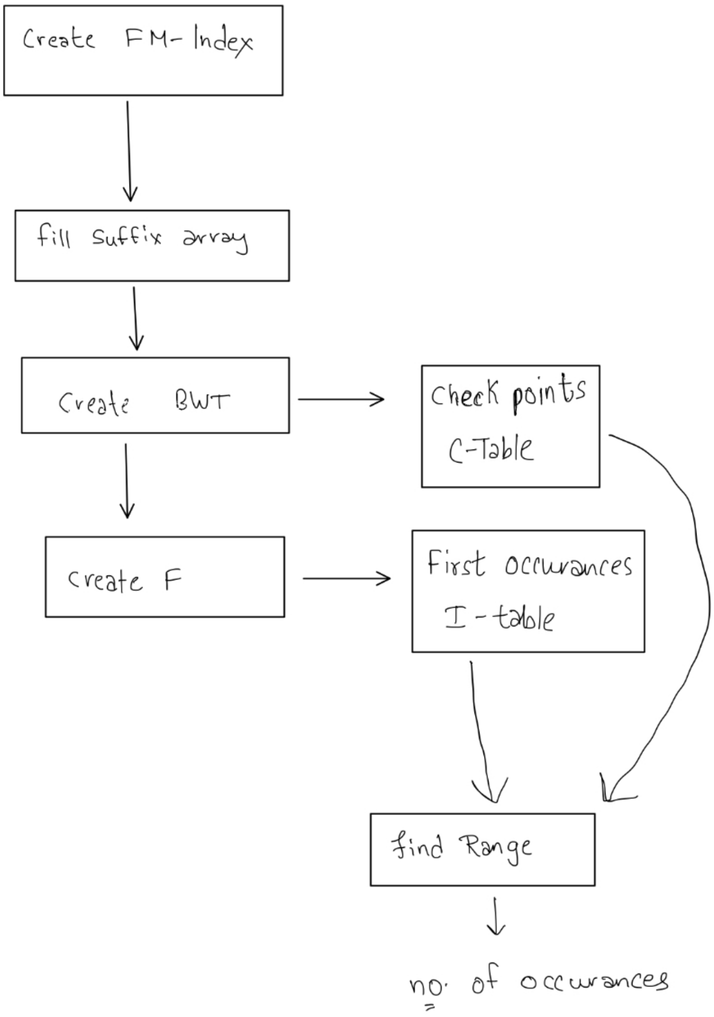
First we have written a c code implements fm-index and finds all the occurrences of a required read(string) in the reference sequence.



```c
    {
        printf("Pattern not found.\n");
        return 0;
    }
    int occurrences = r - l + 1;

    int count = 0;

    for (int p = l; p <= r && count < MAX_OCCURRENCES; p++) {
        positions[count++] = suffix_array[p];

    }

    return occurrences;
}


int main() {
    const char text[MAX_LENGTH] = "ACGTACGTAGCTAGCTAGCTA$";
    const char pattern[] = "TA";
    int pos[MAX_OCCURRENCES] = {0};

    int occ = topfunction(text, pattern, pos);

    if (occ > 0) {
        printf("Occurrences found at positions: ");
        for (int i = 0; i < occ && i < MAX_OCCURRENCES; i++) {
            printf("%d ", pos[i]);
        }
    }
    printf("\n");
    return 0;
}
```

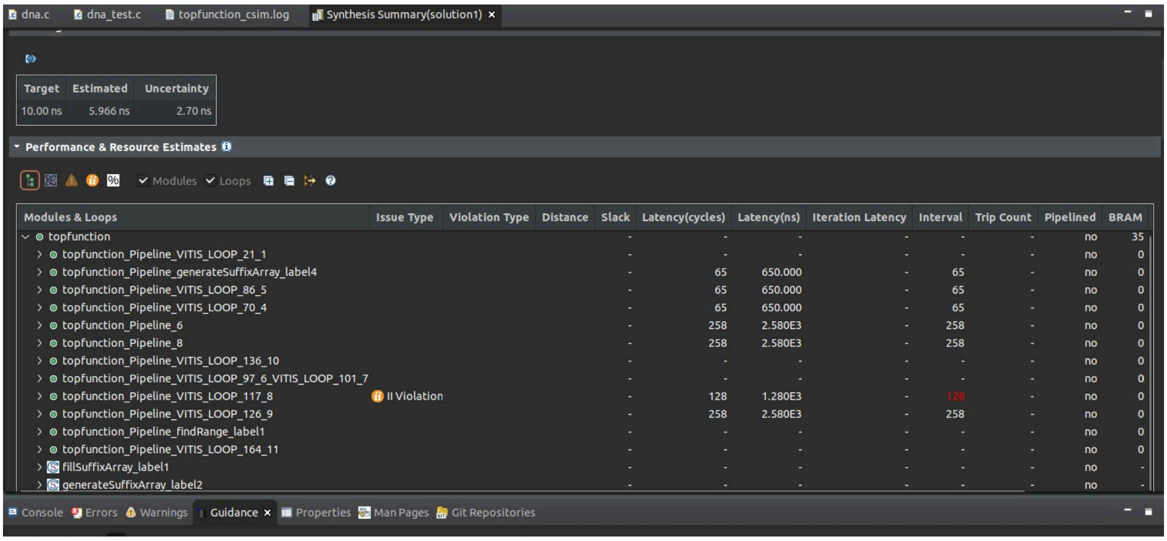
```
Occurrences found at positions: 19 3 15 11 7


...Program finished with exit code 0
Press ENTER to exit console.
```

Block diagram:



Initially we had some timing violations which were gone after using optimisation of the code with pragmas like pipelining and array partitioning.

Latency before optimisation

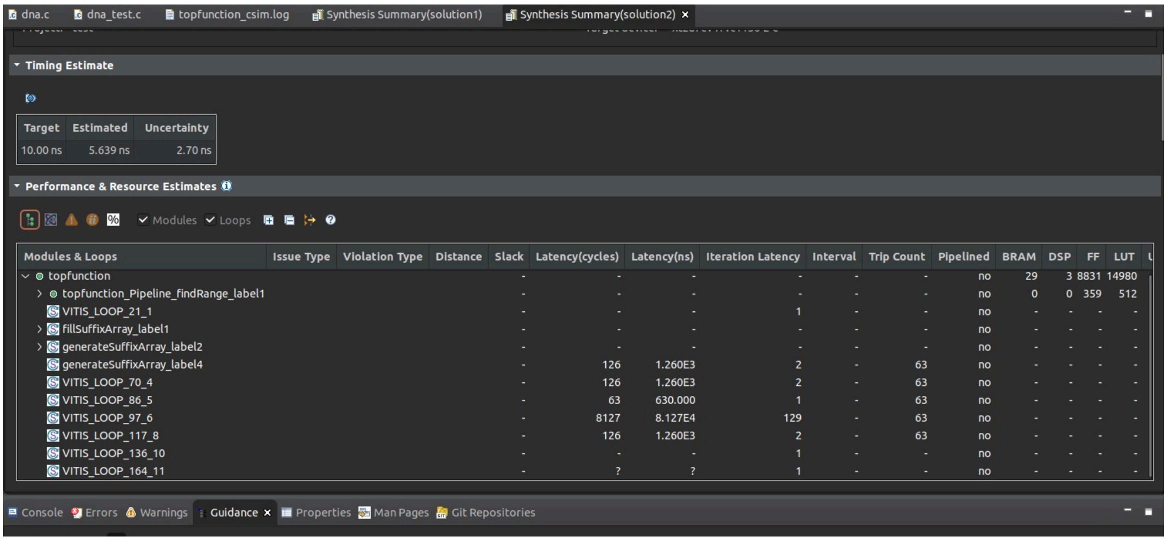


Target	Estimated	Uncertainty
10.00 ns	5.966 ns	2.70 ns

Performance & Resource Estimates

Modules & Loops	Issue Type	Violation Type	Distance	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM
topfunction				-	-	-	-	-	-	no	35
topfunction_Pipeline_VITIS_LOOP_21_1				-	-	-	-	-	-	no	0
topfunction_Pipeline_generateSuffixArray_label4				-	65	650.000	-	65	-	no	0
topfunction_Pipeline_VITIS_LOOP_86_5				-	65	650.000	-	65	-	no	0
topfunction_Pipeline_VITIS_LOOP_70_4				-	65	650.000	-	65	-	no	0
topfunction_Pipeline_6				-	258	2.580E3	-	258	-	no	0
topfunction_Pipeline_8				-	258	2.580E3	-	258	-	no	0
topfunction_Pipeline_VITIS_LOOP_136_10				-	-	-	-	-	-	no	0
topfunction_Pipeline_VITIS_LOOP_97_6_VITIS_LOOP_101_7				-	-	-	-	-	-	no	0
topfunction_Pipeline_VITIS_LOOP_117_8	II Violation			-	128	1.280E3	-	128	-	no	0
topfunction_Pipeline_VITIS_LOOP_126_9				-	258	2.580E3	-	258	-	no	0
topfunction_Pipeline_findRange_label1				-	-	-	-	-	-	no	0
topfunction_Pipeline_VITIS_LOOP_164_11				-	-	-	-	-	-	no	0
fillSuffixArray_label1				-	-	-	-	-	-	no	-
generateSuffixArray_label2				-	-	-	-	-	-	no	-

Latency after optimisation



Timing Estimate

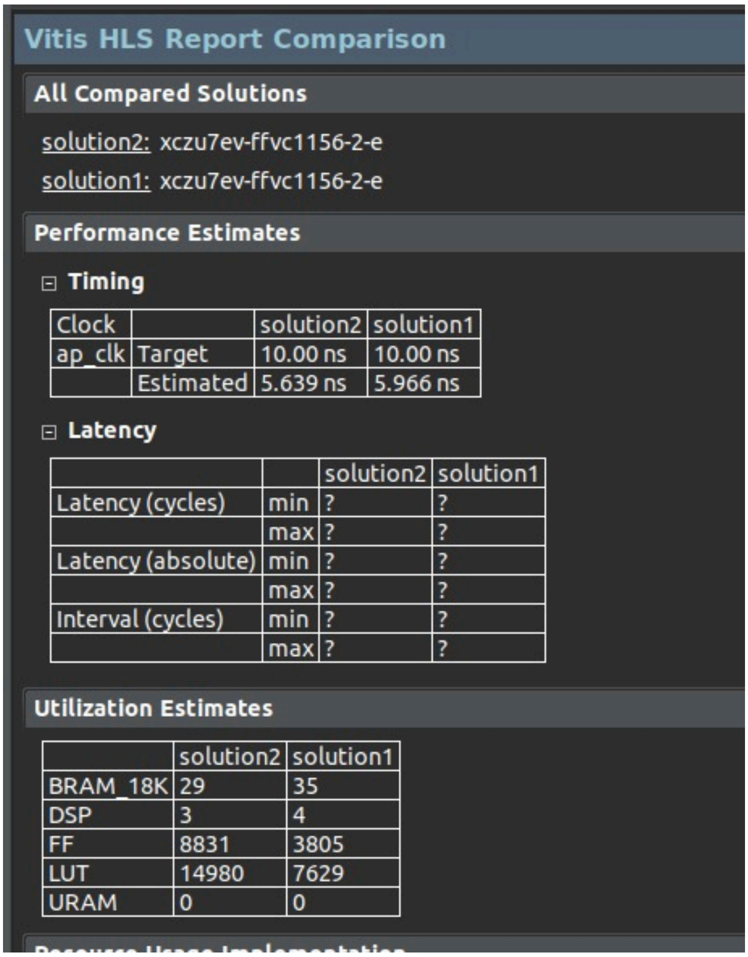
Target	Estimated	Uncertainty
10.00 ns	5.639 ns	2.70 ns

Performance & Resource Estimates

Modules & Loops	Issue Type	Violation Type	Distance	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSP	FF	LUT
topfunction				-	-	-	-	-	-	no	29	3	8831	14980
topfunction_Pipeline_findRange_label1				-	-	-	-	-	-	no	0	0	359	512
VITIS_LOOP_21_1				-	-	-	1	-	-	no	-	-	-	-
fillSuffixArray_label1				-	-	-	-	-	-	no	-	-	-	-
generateSuffixArray_label2				-	-	-	-	-	-	no	-	-	-	-
generateSuffixArray_label4				-	126	1.260E3	2	-	63	no	-	-	-	-
VITIS_LOOP_70_4				-	126	1.260E3	2	-	63	no	-	-	-	-
VITIS_LOOP_86_5				-	63	630.000	1	-	63	no	-	-	-	-
VITIS_LOOP_97_6				-	8127	8.127E4	129	-	63	no	-	-	-	-
VITIS_LOOP_117_8				-	126	1.260E3	2	-	63	no	-	-	-	-
VITIS_LOOP_136_10				-	-	-	1	-	-	no	-	-	-	-
VITIS_LOOP_164_11				-	?	?	1	-	-	no	-	-	-	-

Latency of the circuit is the time taken by the input to generate the output.

We can see the individual latency of each function in the above diagram .So the latency of the circuit after optimization is 5.639 ns.



Vitis HLS Report Comparison

All Compared Solutions

solution2: xczu7ev-ffvc1156-2-e

solution1: xczu7ev-ffvc1156-2-e

Performance Estimates

⊟ Timing

Clock		solution2	solution1
ap_clk	Target	10.00 ns	10.00 ns
	Estimated	5.639 ns	5.966 ns

⊟ Latency

		solution2	solution1
Latency (cycles)	min	?	?
	max	?	?
Latency (absolute)	min	?	?
	max	?	?
Interval (cycles)	min	?	?
	max	?	?

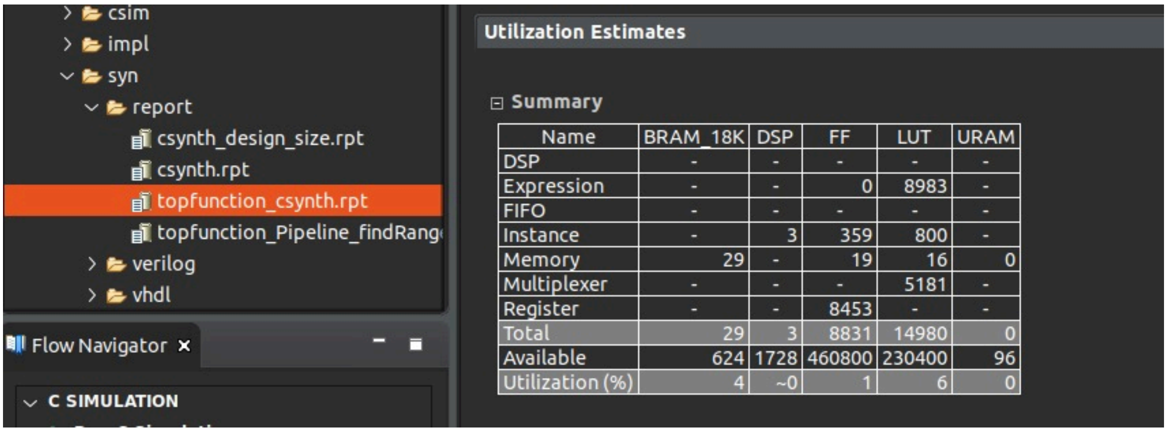
Utilization Estimates

	solution2	solution1
BRAM_18K	29	35
DSP	3	4
FF	8831	3805
LUT	14980	7629
URAM	0	0

Resource Usage Implementation

Comparison report of two solutions

Utilization report of function (findRange)

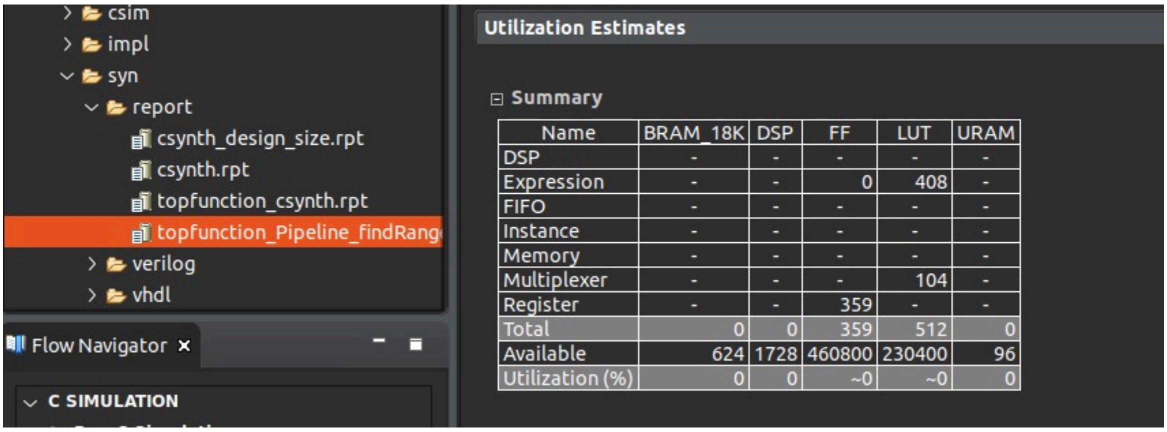


Utilization Estimates

Summary

Name	BRAM_18K	DSP	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	8983	-
FIFO	-	-	-	-	-
Instance	-	3	359	800	-
Memory	29	-	19	16	0
Multiplexer	-	-	-	5181	-
Register	-	-	8453	-	-
Total	29	3	8831	14980	0
Available	624	1728	460800	230400	96
Utilization (%)	4	~0	1	6	0

Utilization report of function (Topfunction)

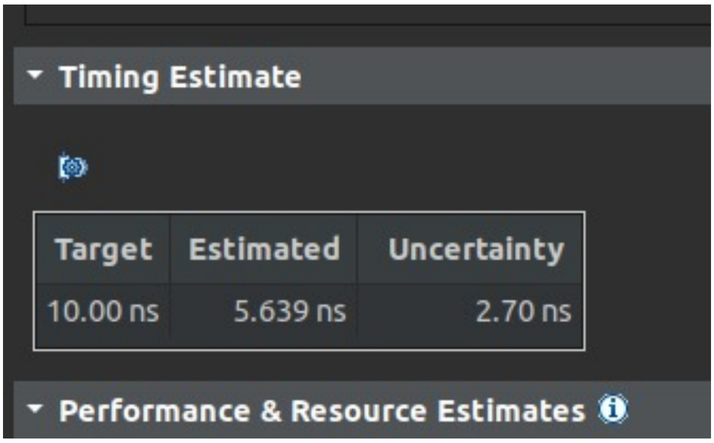


Utilization Estimates

Summary

Name	BRAM_18K	DSP	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	408	-
FIFO	-	-	-	-	-
Instance	-	-	-	-	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	104	-
Register	-	-	359	-	-
Total	0	0	359	512	0
Available	624	1728	460800	230400	96
Utilization (%)	0	0	~0	~0	0

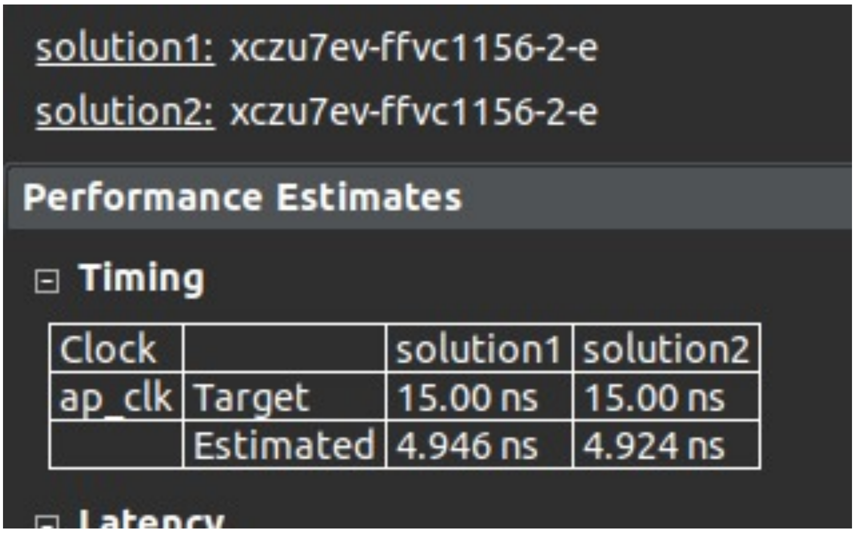
Max Clk Frequency Report



Target	Estimated	Uncertainty
10.00 ns	5.639 ns	2.70 ns

We can that the min clock period is 5.639 ns + 2.70 ns =8.339 ns so the max clock frequency is
1/t_period. So the achieved max frequency is 1/8.339ns = 119.9MHZ.

Using individual block method



solution1: xczu7ev-ffvc1156-2-e

solution2: xczu7ev-ffvc1156-2-e

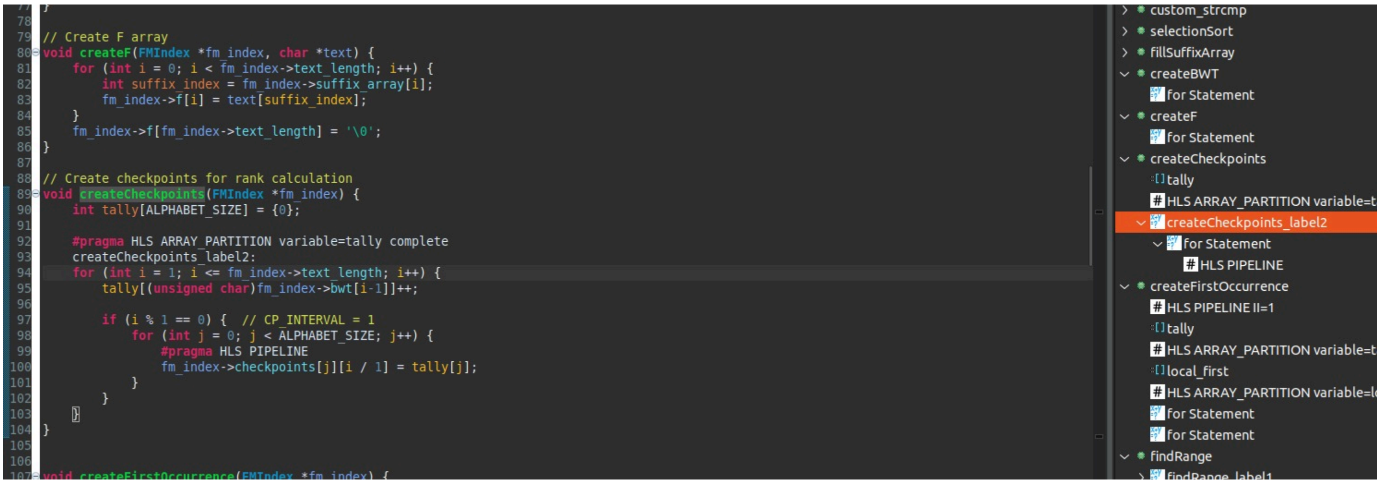
Performance Estimates

⊟ Timing

Clock		solution1	solution2
ap_clk	Target	15.00 ns	15.00 ns
	Estimated	4.946 ns	4.924 ns

⊟ Latency

We can that the min clock period is 4.924 ns + 4.05 ns =8.974 ns   so the max clock frequency is
1/t_period. So the achieved max frequency is 1/8.974 =111.3MHZ.



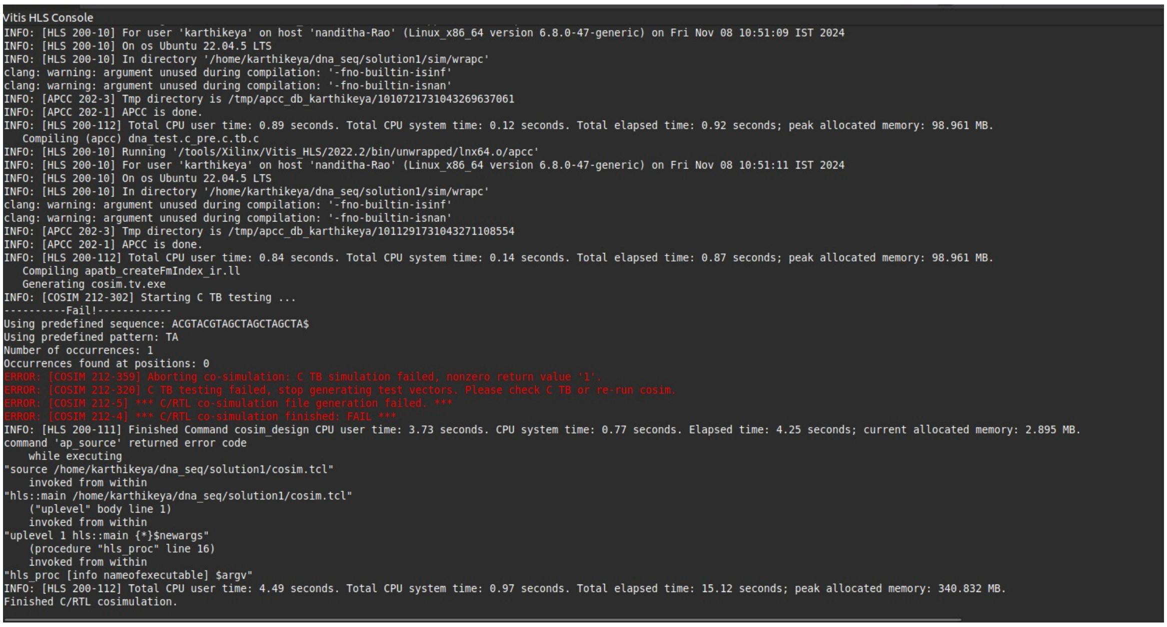
```
}

// Create F array
void createF(FMIndex *fm_index, char *text) {
    for (int i = 0; i < fm_index->text_length; i++) {
        int suffix_index = fm_index->suffix_array[i];
        fm_index->f[i] = text[suffix_index];
    }
    fm_index->f[fm_index->text_length] = '\0';
}

// Create checkpoints for rank calculation
void createCheckpoints(FMIndex *fm_index) {
    int tally[ALPHABET_SIZE] = {0};

    #pragma HLS ARRAY_PARTITION variable=tally complete
    createCheckpoints_label2:
    for (int i = 1; i <= fm_index->text_length; i++) {
        tally[(unsigned char)fm_index->bwt[i-1]]++;

        if (i % 1 == 0) {  // CP_INTERVAL = 1
            for (int j = 0; j < ALPHABET_SIZE; j++) {
                #pragma HLS PIPELINE
                fm_index->checkpoints[j][i / 1] = tally[j];
            }
        }
    }
}


void createFirstOccurrence(FMIndex *fm_index) {
```

```
> custom_strcmp
> selectionSort
> fillSuffixArray
v createBWT
    for Statement
v createF
    for Statement
v createCheckpoints
    tally
    HLS ARRAY_PARTITION variable=t
  v createCheckpoints_label2
    v for Statement
        HLS PIPELINE
v createFirstOccurrence
    HLS PIPELINE II=1
    tally
    HLS ARRAY_PARTITION variable=t
    local_first
    HLS ARRAY_PARTITION variable=l
    for Statement
    for Statement
v findRange
  > findRange_label1
```

Some other pragmas are pipelining in selection sort.

Next we converted it into HLS format and ran it in vitis_hls 2023.2. The board we used for this purpose initially was zybo (xc7z010clg400-1) but we had violations in memory as our task required around 59,000 which were not present in zybo so we resorted to using ZCU104.

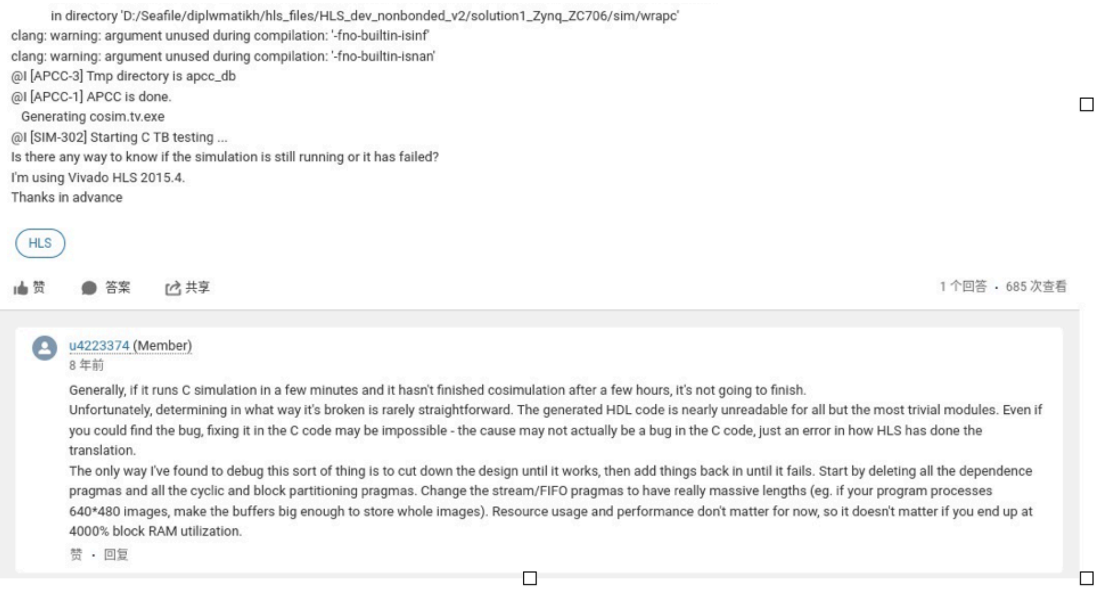
During the process of C/RTL cosimulation we got this error given below



Vitis HLS Console

```
INFO: [HLS 200-10] For user 'karthikeya' on host 'nanditha-Rao' (Linux_x86_64 version 6.8.0-47-generic) on Fri Nov 08 10:51:09 IST 2024
INFO: [HLS 200-10] On os Ubuntu 22.04.5 LTS
INFO: [HLS 200-10] In directory '/home/karthikeya/dna_seq/solution1/sim/wrapc'
clang: warning: argument unused during compilation: '-fno-builtin-isinf'
clang: warning: argument unused during compilation: '-fno-builtin-isnan'
INFO: [APCC 202-3] Tmp directory is /tmp/apcc_db_karthikeya/101072173104326963706l
INFO: [APCC 202-1] APCC is done.
INFO: [HLS 200-112] Total CPU user time: 0.89 seconds. Total CPU system time: 0.12 seconds. Total elapsed time: 0.92 seconds; peak allocated memory: 98.961 MB.
   Compiling (apcc) dna_test.c_pre.c.tb.c
INFO: [HLS 200-10] Running '/tools/Xilinx/Vitis_HLS/2022.2/bin/unwrapped/lnx64.o/apcc'
INFO: [HLS 200-10] For user 'karthikeya' on host 'nanditha-Rao' (Linux_x86_64 version 6.8.0-47-generic) on Fri Nov 08 10:51:11 IST 2024
INFO: [HLS 200-10] On os Ubuntu 22.04.5 LTS
INFO: [HLS 200-10] In directory '/home/karthikeya/dna_seq/solution1/sim/wrapc'
clang: warning: argument unused during compilation: '-fno-builtin-isinf'
clang: warning: argument unused during compilation: '-fno-builtin-isnan'
INFO: [APCC 202-3] Tmp directory is /tmp/apcc_db_karthikeya/101129173104327110855 4
INFO: [APCC 202-1] APCC is done.
INFO: [HLS 200-112] Total CPU user time: 0.84 seconds. Total CPU system time: 0.14 seconds. Total elapsed time: 0.87 seconds; peak allocated memory: 98.961 MB.
   Compiling apatb_createFmIndex_ir.ll
   Generating cosim.tv.exe
INFO: [COSIM 212-302] Starting C TB testing ...
----------Fail!------------
Using predefined sequence: ACGTACGTAGCTAGCTAGCTA$
Using predefined pattern: TA
Number of occurrences: 1
Occurrences found at positions: 0
ERROR: [COSIM 212-359] Aborting co-simulation: C TB simulation failed, nonzero return value '1'.
ERROR: [COSIM 212-320] C TB testing failed, stop generating test vectors. Please check C TB or re-run cosim.
ERROR: [COSIM 212-5] *** C/RTL co-simulation file generation failed. ***
ERROR: [COSIM 212-4] *** C/RTL co-simulation finished: FAIL ***
INFO: [HLS 200-111] Finished Command cosim_design CPU user time: 3.73 seconds. CPU system time: 0.77 seconds. Elapsed time: 4.25 seconds; current allocated memory: 2.895 MB.
command 'ap_source' returned error code
    while executing
"source /home/karthikeya/dna_seq/solution1/cosim.tcl"
    invoked from within
"hls::main /home/karthikeya/dna_seq/solution1/cosim.tcl"
    ("uplevel" body line 1)
    invoked from within
"uplevel 1 hls::main {*}$newargs"
    (procedure "hls_proc" line 16)
    invoked from within
"hls_proc [info nameofexecutable] $argv"
INFO: [HLS 200-112] Total CPU user time: 4.49 seconds. Total CPU system time: 0.97 seconds. Total elapsed time: 15.12 seconds; peak allocated memory: 340.832 MB.
Finished C/RTL cosimulation.
```

The test bench was working properly for c-simulation and c-synthesis but for c/rtl cosimulation it was able to synthesise the suffix_array so we tried implementation of each module in the code seperately to see if they are synthesizeable or not and have updated parts which were not syntesizeable like (while loops , pointers , struct). Even then we weren't able to make the rtl co-simulation work It went on running for hours. We had discussed with our Mentor and tried debugging it but we weren't able to do it.



```
        in directory 'D:/Seafile/diplwmatikh/hls_files/HLS_dev_nonbonded_v2/solution1_Zynq_ZC706/sim/wrapc'
clang: warning: argument unused during compilation: '-fno-builtin-isinf'
clang: warning: argument unused during compilation: '-fno-builtin-isnan'
@I [APCC-3] Tmp directory is apcc_db
@I [APCC-1] APCC is done.
  Generating cosim.tv.exe
@I [SIM-302] Starting C TB testing ...
```

Is there any way to know if the simulation is still running or it has failed?
I'm using Vivado HLS 2015.4.
Thanks in advance

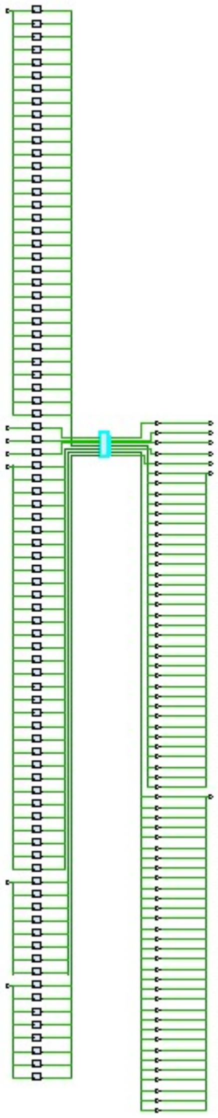
u4223374 (Member)
8 年前

Generally, if it runs C simulation in a few minutes and it hasn't finished cosimulation after a few hours, it's not going to finish.

Unfortunately, determining in what way it's broken is rarely straightforward. The generated HDL code is nearly unreadable for all but the most trivial modules. Even if you could find the bug, fixing it in the C code may be impossible - the cause may not actually be a bug in the C code, just an error in how HLS has done the translation.

The only way I've found to debug this sort of thing is to cut down the design until it works, then add things back in until it fails. Start by deleting all the dependence pragmas and all the cyclic and block partitioning pragmas. Change the stream/FIFO pragmas to have really massive lengths (eg. if your program processes 640*480 images, make the buffers big enough to store whole images). Resource usage and performance don't matter for now, so it doesn't matter if you end up at 4000% block RAM utilization.

Layout design



Github Link to the code:

https://github.com/Vadlamudi04/dna_sequencing

References:

1. https://www.cs.jhu.edu/~langmea/resources/lecture_notes/bwt_and_fm_index.pdf
2. https://ieeexplore.ieee.org/abstract/document/5771277?casa_token=c6pxv7GcrcMAAAAA:YXWEw9T2NoYs4aol0DbKkq9KaoofpqFIeP-f5FM6F2-SYKsFChiRF5BmcTBJxsJip196N_gQEro
3. https://github.com/BenLangmead/comp-genomics-class/tree/master
4. P. Ferragina and G. Manzini, "Opportunistic data structures with applications," in Proc. 41st Annu. Symp. Found. Comput. Sci., 2000, pp. 390–398.