PROJECT REPORT FOR GROUP NO 15

SPAM FILTER

Guided by:

Prof. Chandan Karfa

(Assistant professor IIT Guwahati)

Mentor:

Debabrata Senapati

(PhD Computer Science, IIT Guwahati)

GROUP MEMBERS:

• Amin Dhruvil Rameshchandra	204101006
• Ankit Kumar Jatiya	204101008
• Chetan Pralhad Ingle	204101019
• Jeshwanth Naladala	204101036
• Ronak Chabukswar	204101047

Description of Project:

The design trains a logistic regression model to classify emails as either spam or not spam. We use stochastic gradient descent in our accelerator to train the model.

Objective of the project:

The main objective of the project is to optimize the given area and latency required in the spam filtering application given to us using various optimization techniques taught in the lectures by sir.

Phase I

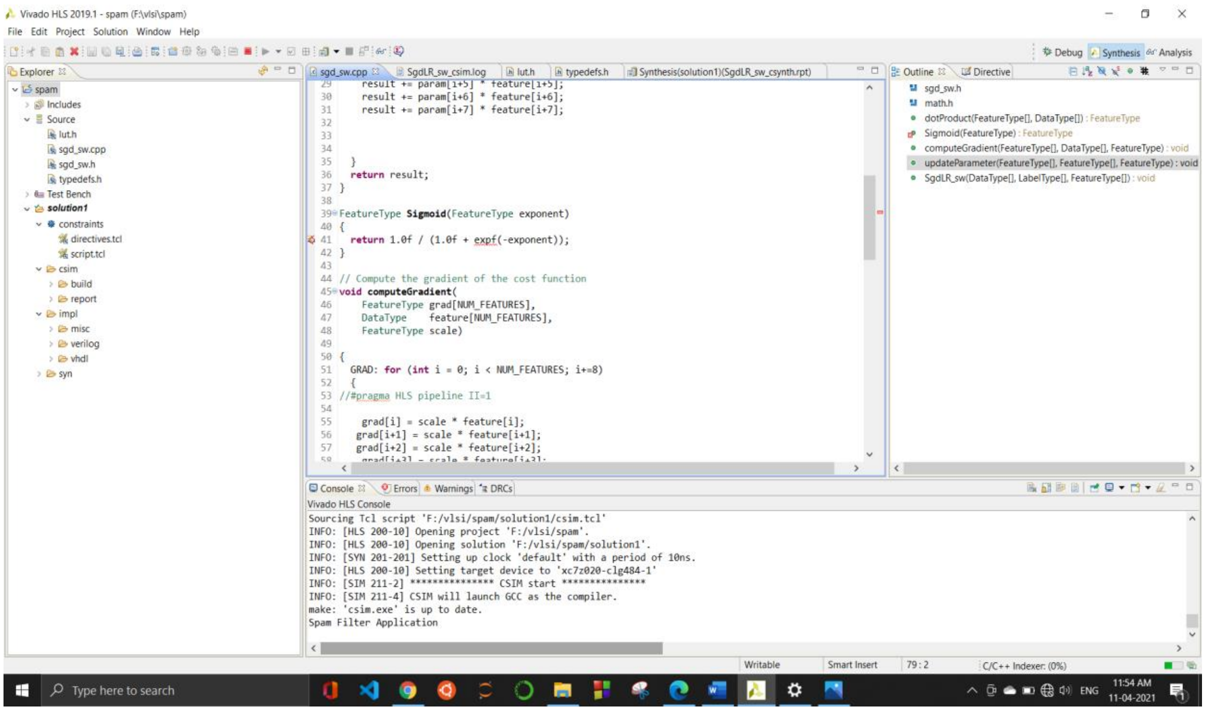
In 1st phase of the project we tried to understand the code of the project. And after that we run the code.

Tools used:

The tool we have used for this project is **XILINX VIVADO HLS 2019.1**.

Steps to run the Application:

- Start the XILINX VIVADO HLS 2019.1 tool first.
- Create new project.
- Add the proper path of project and name the project accordingly.
- First add all the source file given in project in the ADD/REMOVE file pop up.
- Use the top function as SgdLR_sw.
- Use all the host folder files for creating the testbench.
- Also remember to add shufflelabels.dat and shuffledeats.dat files in testbench.
- In the solution configuration use the clock period as 10.
- Select the xc7z020clg484-1 from the browse option.
- Select the device board as zedBoardzynq Evaluation and Development kit.
- Run the C simulation to view the expected output.
- Select project->Run C simulation and click OK.
- After that click on the run c synthesis to start the synthesis.

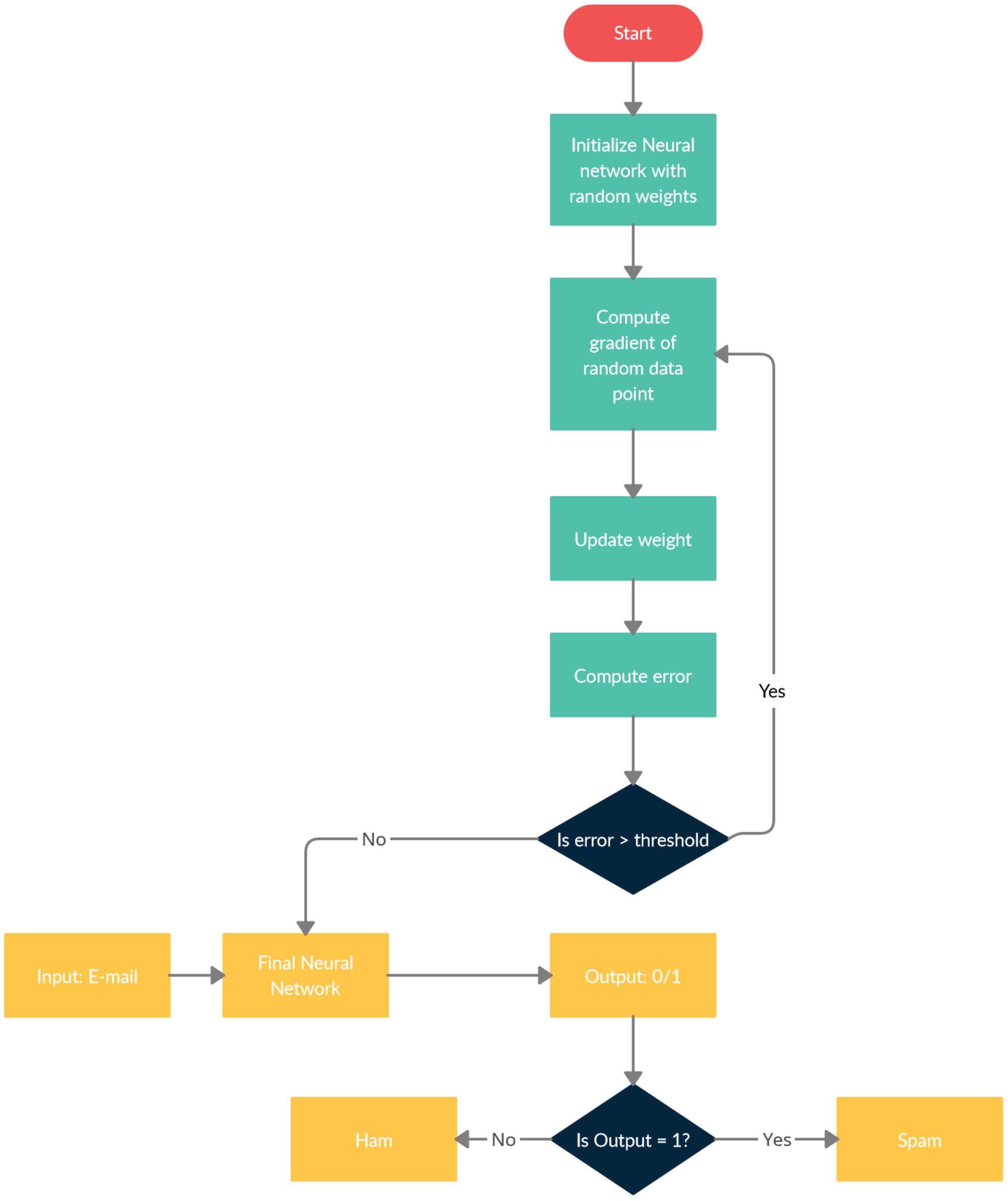

Vivado HLS 2019.1 - spam (F:\vlsi\spam)
File Edit Project Solution Window Help
Debug
Synthesis
Analysis
Explorer
spam
Includes
Source
lut.h
sgd_sw.cpp
sgd_sw.h
typedefs.h
Test Bench
solution1
constraints
directives.tcl
script.tcl
csim
build
report
impl
misc
verilog
vhdl
syn
sgd_sw.cpp
SgdLR_sw_csim.log
lut.h
typedefs.h
Synthesis(solution1)(SgdLR_sw_csynth.rpt)
Outline
Directive
sgd_sw.h
math.h
dotProduct(FeatureType[], DataType[]) : FeatureType
Sigmoid(FeatureType) : FeatureType
computeGradient(FeatureType[], DataType[], FeatureType) : void
updateParameter(FeatureType[], FeatureType[], FeatureType) : void
SgdLR_sw(DataType[], LabelType[], FeatureType[]) : void
Console
Errors
Warnings
DRCs
Vivado HLS Console
Writable
Smart Insert
79 : 2
C/C++ Indexer: (0%)
Type here to search
ENG
11:54 AM
11-04-2021

```
    result += param[i+5] * feature[i+5];
    result += param[i+6] * feature[i+6];
    result += param[i+7] * feature[i+7];



  }
  return result;
}

FeatureType Sigmoid(FeatureType exponent)
{
  return 1.0f / (1.0f + expf(-exponent));
}

// Compute the gradient of the cost function
void computeGradient(
    FeatureType grad[NUM_FEATURES],
    DataType    feature[NUM_FEATURES],
    FeatureType scale)

{
  GRAD: for (int i = 0; i < NUM_FEATURES; i+=8)
  {
//#pragma HLS pipeline II=1

    grad[i] = scale * feature[i];
   grad[i+1] = scale * feature[i+1];
   grad[i+2] = scale * feature[i+2];
```

```
Sourcing Tcl script 'F:/vlsi/spam/solution1/csim.tcl'
INFO: [HLS 200-10] Opening project 'F:/vlsi/spam'.
INFO: [HLS 200-10] Opening solution 'F:/vlsi/spam/solution1'.
INFO: [SYN 201-201] Setting up clock 'default' with a period of 10ns.
INFO: [HLS 200-10] Setting target device to 'xc7z020-clg484-1'
INFO: [SIM 211-2] *************** CSIM start ***************
INFO: [SIM 211-4] CSIM will launch GCC as the compiler.
make: 'csim.exe' is up to date.
Spam Filter Application
```

Code Flow:

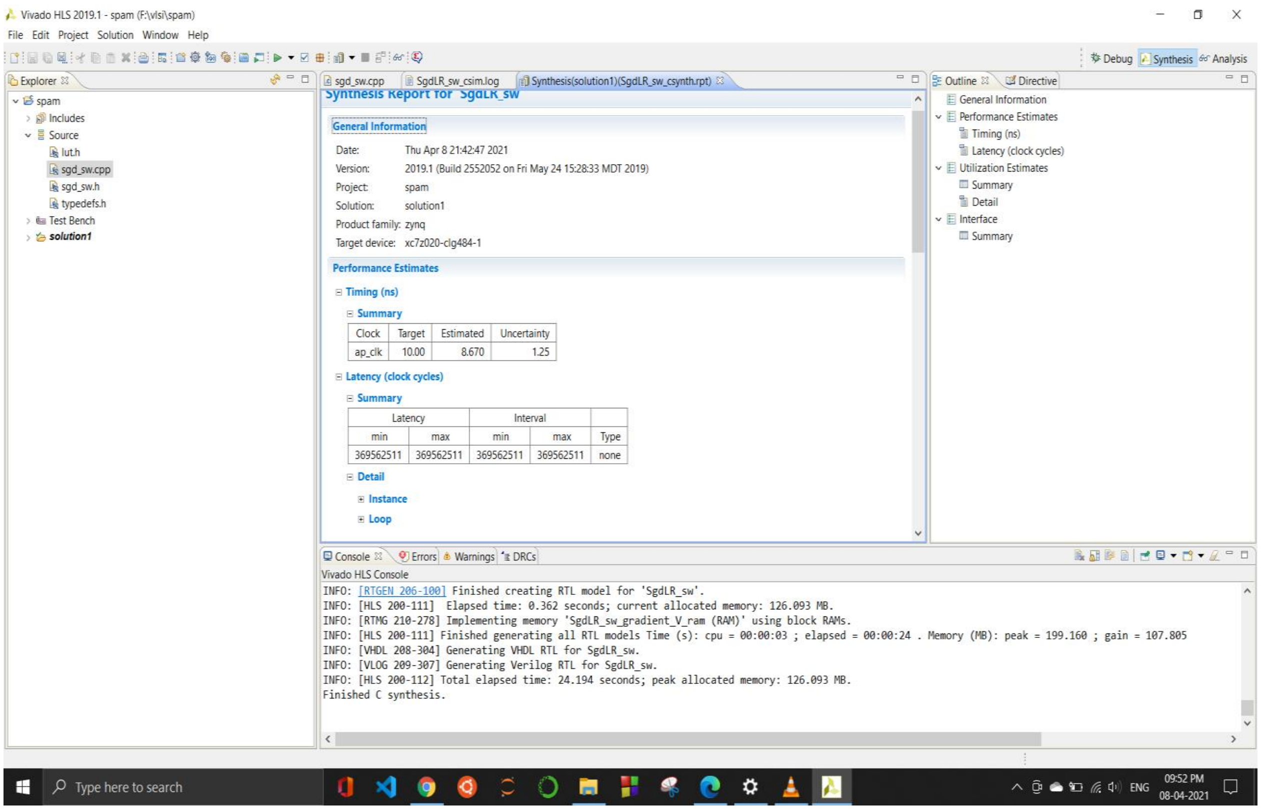
The code flow of the code which was assigned to us is given below in form of diagram.



Phase II

The approach we have followed in phase II is given below.

- First, we analyze all the results obtained after running the application
- We noted down all the latency and utilization of the resources in our project.



Synthesis Report for 'SgdLR_sw'

General Information

Date:	Thu Apr 8 21:42:47 2021
Version:	2019.1 (Build 2552052 on Fri May 24 15:28:33 MDT 2019)
Project:	spam
Solution:	solution1
Product family:	zynq
Target device:	xc7z020-clg484-1

Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	8.670	1.25

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
369562511	369562511	369562511	369562511	none

Detail

Instance

Loop

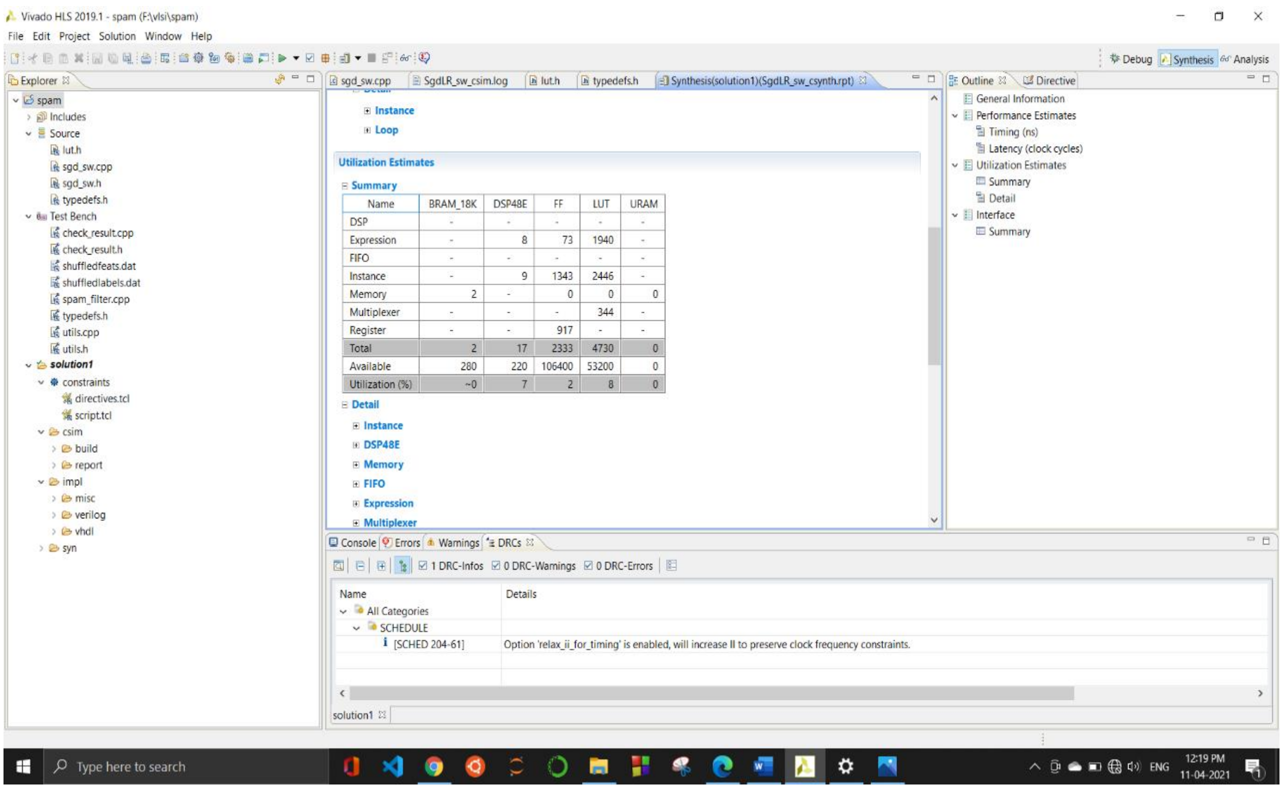
Vivado HLS Console

```
INFO: [RTGEN 206-100] Finished creating RTL model for 'SgdLR_sw'.
INFO: [HLS 200-111]  Elapsed time: 0.362 seconds; current allocated memory: 126.093 MB.
INFO: [RTMG 210-278] Implementing memory 'SgdLR_sw_gradient_V_ram (RAM)' using block RAMs.
INFO: [HLS 200-111] Finished generating all RTL models Time (s): cpu = 00:00:03 ; elapsed = 00:00:24 . Memory (MB): peak = 199.160 ; gain = 107.805
INFO: [VHDL 208-304] Generating VHDL RTL for SgdLR_sw.
INFO: [VLOG 209-307] Generating Verilog RTL for SgdLR_sw.
INFO: [HLS 200-112] Total elapsed time: 24.194 seconds; peak allocated memory: 126.093 MB.
Finished C synthesis.
```

The latency we got from the spam filter code is:

Minimum latency: 369562511

Maximum latency: 369562511



Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	8	73	1940	-
FIFO	-	-	-	-	-
Instance	-	9	1343	2446	-
Memory	2	-	0	0	0
Multiplexer	-	-	-	344	-
Register	-	-	917	-	-
Total	2	17	2333	4730	0
Available	280	220	106400	53200	0
Utilization (%)	~0	7	2	8	0

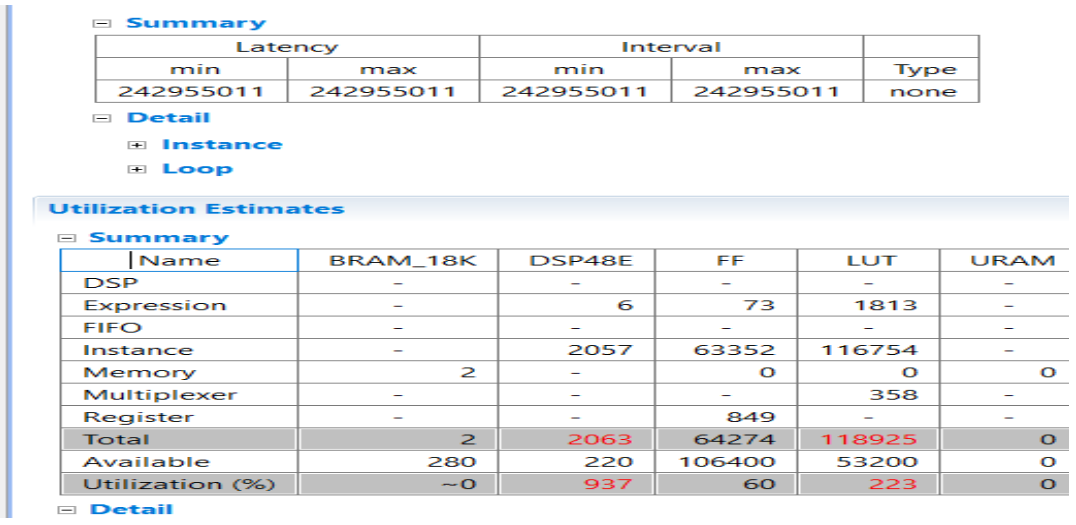
```
Name                Details
All Categories
  SCHEDULE
    [SCHED 204-61]  Option 'relax_ii_for_timing' is enabled, will increase II to preserve clock frequency constraints.
```

Attempt for Latency:

- **Full loop unrolling along with partial unrolling.**

We have applied the pipeline pragma in the two for loops where each of the loop has predefined static number of iteration bound. Number of features is the constrained given in the loop which is 1024. So, we tried to apply complete loop unroll on one of the loops, but in the result, we got to know that that the resources are going out of bound. The results are shown below.

Command used in for loop.
#pragma HLS unroll



Summary

Latency		Interval		
min	max	min	max	Type
242955011	242955011	242955011	242955011	none

Detail

Instance

Loop

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	6	73	1813	-
FIFO	-	-	-	-	-
Instance	-	2057	63352	116754	-
Memory	2	-	0	0	0
Multiplexer	-	-	-	358	-
Register	-	-	849	-	-
Total	2	2063	64274	118925	0
Available	280	220	106400	53200	0
Utilization (%)	~0	937	60	223	0

Detail

- **Partial loop unrolling with various factors.**

  We have tried to apply the partial loop unrolling on the loops with the factor of 2,4,8,16, etc and tried to analyze the results.
  The results are shown below. The unroll factor is decided manually by starting from 2. We have used the pragma command to unroll the loop which is given below.
  #pragma unroll factor=2
  #pragma unroll factor=4
  #pragma unroll factor=8
  .
  .
  .
  We have to stop at the unroll factor of 32. Since the resources were going out of bound at 32 factor. After that there is no meaning in unrolling the loop further by any of the higher factor.

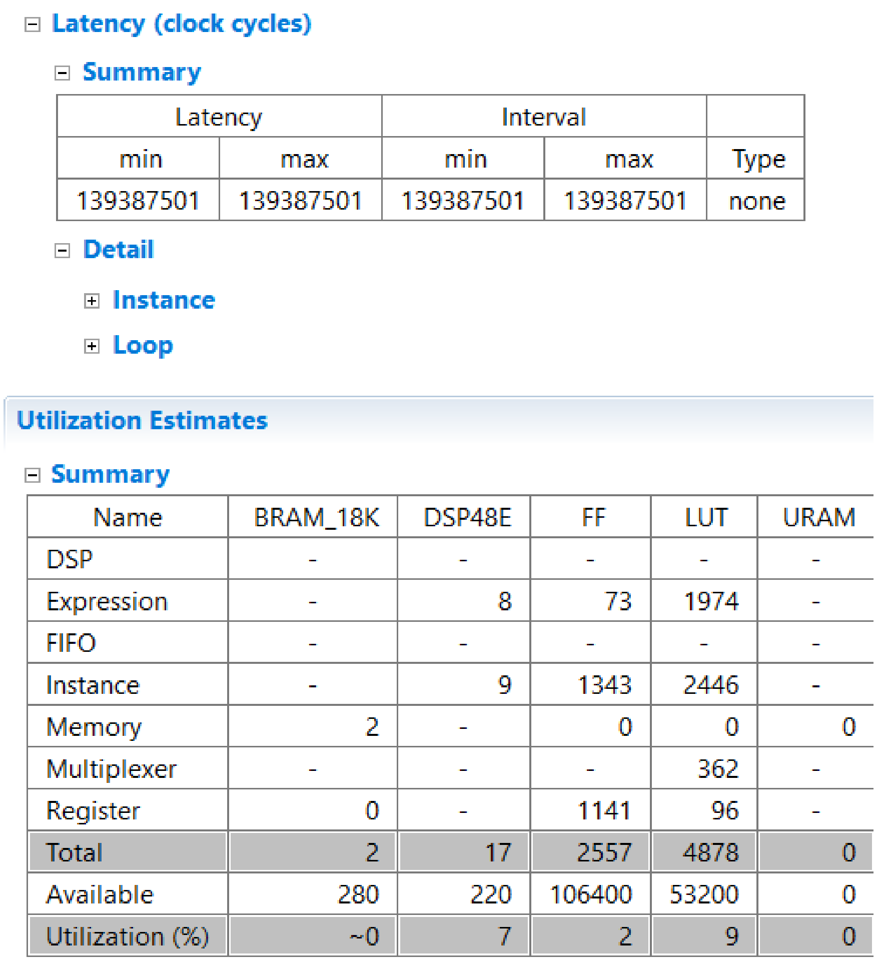
Attempt for area:

There is always a trade off between area and latency. We can't reduce both the attributes simultaneously. So, we decided to optimize the area according to the optimal latency we have got from the above results.

We have got optimal latency using two loops unroll by factor of 32 + one loop unroll by factor of 16. But the utilization of resources was very much high. So, by intuition and the given methods we tried to optimize the area as well as latency simultaneously to get the balance between them.
All the results we have got are given below. And the optimal latency and area is also given.

Area optimization attempt:

- We have got the optimal latency using the methods applied above.
- But the resource utilization was increasing rapidly with such a low latency.
- We tried to apply 2 pragma pipelines for the two loops and we got the following result.
- It shows that the area was reduced at very much extent, rather it is the optimal area we have got but the only problem occurred here was the latency.
- Optimal latency value we have got earlier was 54292511. But after applying the 2-pipeline method we have got the increased latency which is 139387501.



⊟ **Latency (clock cycles)**

⊟ **Summary**

Latency		Interval		
min	max	min	max	Type
139387501	139387501	139387501	139387501	none

⊟ **Detail**

⊞ **Instance**

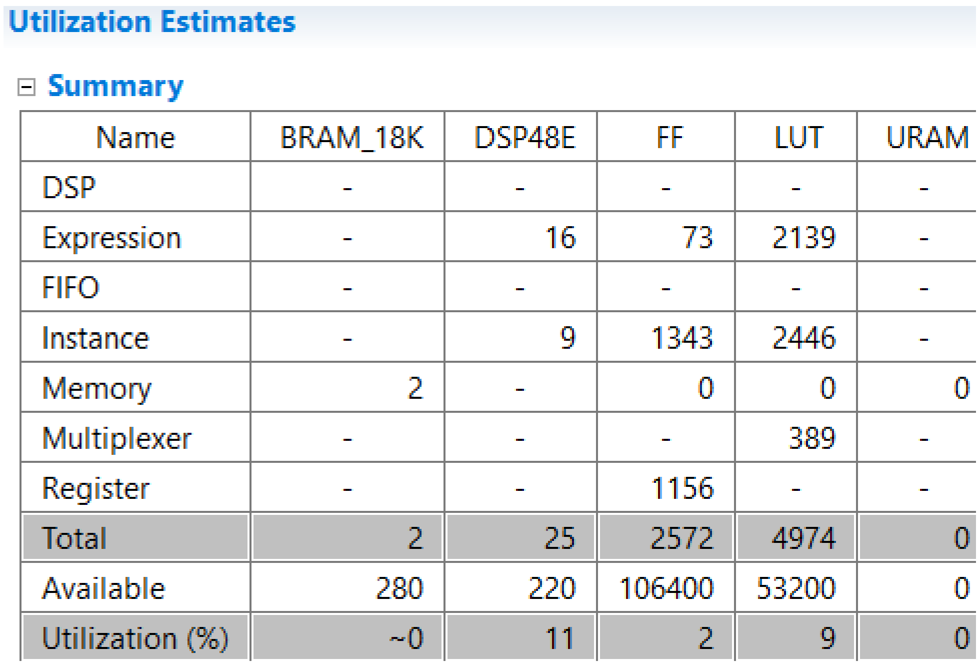
⊞ **Loop**

Utilization Estimates

⊟ **Summary**

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	8	73	1974	-
FIFO	-	-	-	-	-
Instance	-	9	1343	2446	-
Memory	2	-	0	0	0
Multiplexer	-	-	-	362	-
Register	0	-	1141	96	-
Total	2	17	2557	4878	0
Available	280	220	106400	53200	0
Utilization (%)	~0	7	2	9	0

- We tried one more attempt towards reducing the area in such a way that latency should not increase drastically.
- We have applied 3 partial loops unrolling with the factor of 2.
- We got the more or less same results which we got earlier.
- The results are given as follows.



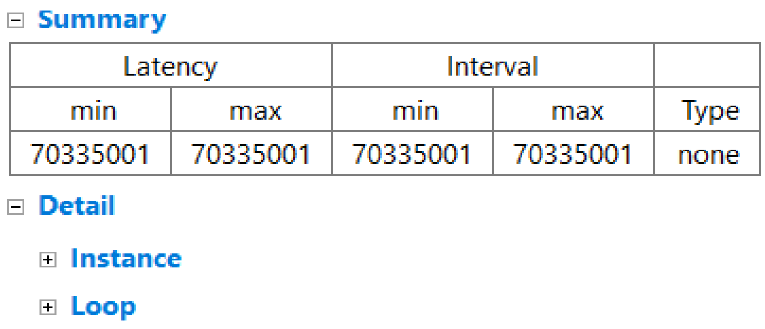
Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	16	73	2139	-
FIFO	-	-	-	-	-
Instance	-	9	1343	2446	-
Memory	2	-	0	0	0
Multiplexer	-	-	-	389	-
Register	-	-	1156	-	-
Total	2	25	2572	4974	0
Available	280	220	106400	53200	0
Utilization (%)	~0	11	2	9	0

- Finally, we have tried to attempt the 3 pragma HLS pipeline on the loops.
- We got some surprising results in which the latency was also reduced to almost 80 % and the area was also utilized minimum among all the methods we have tried so far.
- The result is given as follows.

Optimal Area Result:



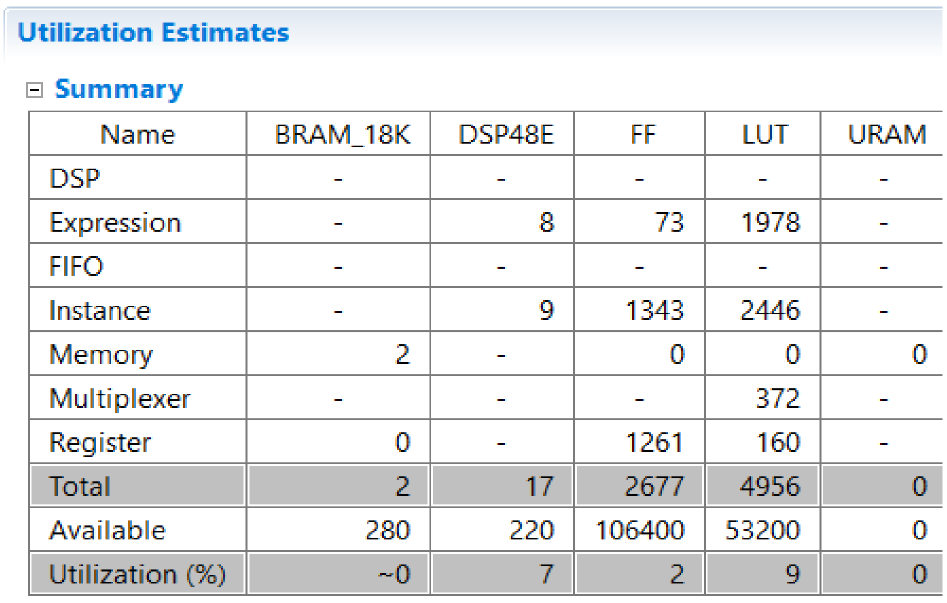
Summary

Latency		Interval		
min	max	min	max	Type
70335001	70335001	70335001	70335001	none

Detail

Instance

Loop

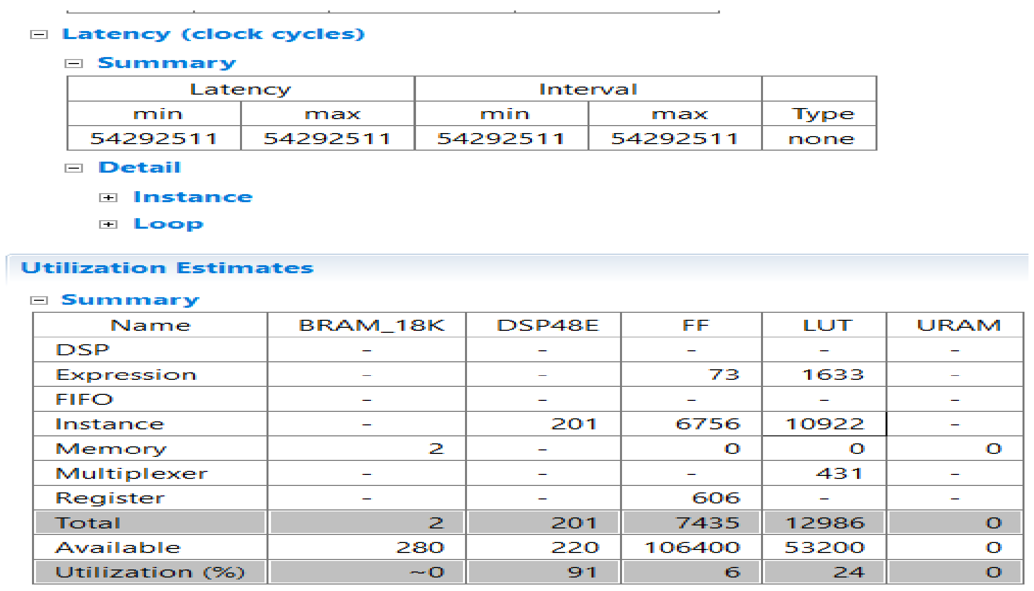


Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	8	73	1978	-
FIFO	-	-	-	-	-
Instance	-	9	1343	2446	-
Memory	2	-	0	0	0
Multiplexer	-	-	-	372	-
Register	0	-	1261	160	-
Total	2	17	2677	4956	0
Available	280	220	106400	53200	0
Utilization (%)	~0	7	2	9	0

Optimal Latency Result:


Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
54292511	54292511	54292511	54292511	none

Detail

Instance

Loop

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	73	1633	-
FIFO	-	-	-	-	-
Instance	-	201	6756	10922	-
Memory	2	-	0	0	0
Multiplexer	-	-	-	431	-
Register	-	-	606	-	-
Total	2	201	7435	12986	0
Available	280	220	106400	53200	0
Utilization (%)	~0	91	6	24	0

Final Results and Analysis:

Table 1: Latency

Optimization	Latency
No optimization	369562511
2 pragma HLS pipeline	139387501
3 partial loop unroll (factor of 2)	185242411
3 partial loop unroll (factor of 4)	110362511
3 partial loop unroll (factor of 8)	75847511
3 partial loop unroll (factor of 16)	61492511
3 partial loop unroll (factor of 32)	54292511
2 partial loop unroll(32) + 1 HLS pipeline	54315001
3 pragma HLS pipeline	70335001
2 pragma HLS pipeline + 1 partial loop unroll	70290001
partial loop unroll(32) + 1 partial(16)	54292511

Table 2: Area

Optimization	Resources				
	BRAM_18K	DSP48E	FF	LUT	URAM
No optimization	2	17	2333	4730	0
3 pragma HLS pipeline	2	17	2677	4956	0
3 partial loop unroll (factor of 2)	2	25	2572	4974	0
3 partial loop unroll (factor of 4)	2	41	2953	5516	0
3 partial loops unroll (factor of 8)	2	73	4083	7002	0

3 partial loop unroll (factor of 16)	2	137	5723	9746	0
3 partial loop unroll (factor of 32)	2	265	8891	14750	0
2 partial loop unroll(32) + 1 HLS pipeline	2	141	5997	11300	0
partial loop unroll(32) + 1 partial(16)	2	201	7435	12986	0
2 pragma HLS pipeline + 1 partial loop unroll	2	45	3442	5778	0
2 pragma HLS pipeline	2	17	2557	4878	0

🟨 -----	optimal
🟩 -----	trade-off
🟥 -----	resources out of bounds

Conclusion:

We are able to reduce the latency up to 85.4 % using the optimization techniques. But after considering the area and latency trade-off we have also optimized the area in such a way that latency is optimized up to 81%.

This is the best result we have got from the analysis we have done on the project given to us.