Assignment 1
ECE1373S: Digital Design for Systems on Chip
May 4, 2015

Arjun Rakesh Gandhi

Student Number: 1001070119

Index

List of Figures

Problem statement: In this assignment, a module for the classifier layer used in neural networks using high level synthesis in Vivado HLS was generated and then integrated with a microblaze controller in Vivado.

I. Background:

The main layers in a convolutional neural network are:

1). Convolutional layer

2). LRN layer (Local response normalization)

3). Pooling layer

4). Classifier layer

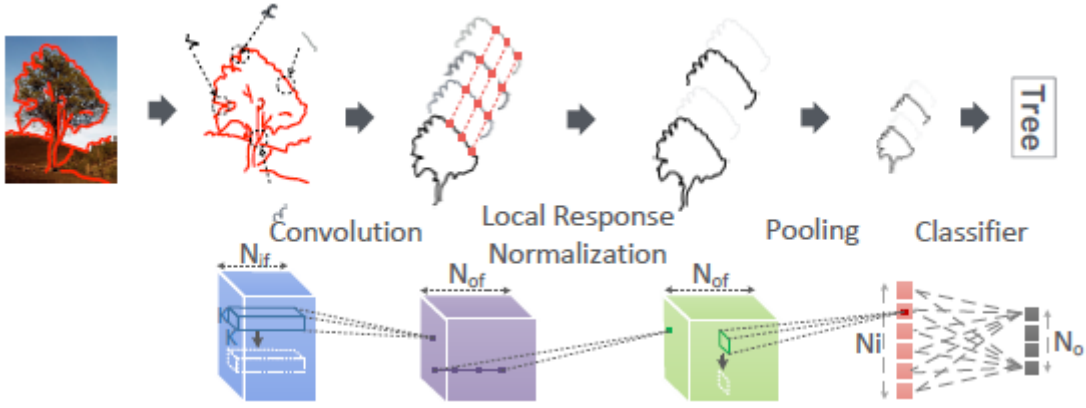

Figure 1 Example of a neural network design

This assignment is focused on implementation of a classifier layer. In a simpler version, this layer takes in inputs which are outputs from neurons of the preceding layer as can be seen in the figure. The output of each neuron is multiplied by a corresponding weight which is set during the training phase (off line or on line training) and given to each neuron of the classifier layer. The classifier sums these weighted inputs and passes them through a predefined function which basically classifies the output to one of the pre-defined values. In this module, the weighted inputs will be added together and then added to a fixed bias value. If the final sum is greater than a threshold value (pre-determined), then the output will be logical 1 or else it will be logical 0.

CODE: The code for the module is shown as follows:

```
void classifier(float in_array[NO_IN_NODES], float synaptic_weights[NO_OUT_NODES *
NO_IN_NODES], float out_array[NO_OUT_NODES)
{
	//*************************************************************************
	// If start is set and reset is low, start the functionality.
	// This logic will take in input from all input neurons at all the output nodes,
	// consider their weighted sum and after adding bias, will evaluate against
	// threshold and give output.
	//*************************************************************************
	//-------------------------------------------------------------------
	Output_node_loop: for (int i = 0; i < NO_OUT_NODES; i++)
		{
			out_array[i] = 0;
	//=================================================================
		weighted_sum_loop: for (int j = 0; j < NO_IN_NODES; j++)
			{
			out_array[i] += in_array[j] * synaptic_weights[(i*NO_IN_NODES) + j];
			}
	//=================================================================
			if ((out_array[i] + BIAS) >= THRESHOLD)
			{
				out_array[i] = 1;
			}
			else
			{
				out_array[i] = 0;
			}
	//=================================================================
		} //	for (int i = 0; i < NO_OUT_NODES; i++)
} // void classifier(float in_array[NO_IN_NODES],
```

II. Input/Outputs:

The interfaces for the input array (representing output of preceding layer), the synaptic weights array as well as that of the output array are AXI Lite interfaces. The reason for choosing AXI Lite is easily incorporation of the module in a microblaze run environment.

III. Test Bench

In the actual CNN (Convolutional neural network), the classifier layer will be the last layer. But to test this module on a standalone basis, a test bench will provide it with inputs and synaptic weights and will monitor the outputs generated. As inputs are predefined along with the weights, the outputs are also known beforehand. The outputs from the module are compared to the predetermined outputs and if they match, the test passes and it can be determined that the calculations done by the module are correct. C simulation result indicating success of the test case is as follows:

```
Starting C simulation ...

C:/Xilinx/Vivado_HLS/2014.1/bin/vivado_hls.bat
C:/Users/Ag/Documents/HLS/Classifier_layer/classifier/axi_wo_ar_part/csim.tcl
@I [LIC-101] Checked out feature [VIVADO_HLS]
@I [HLS-10] Running
'C:/Xilinx/Vivado_HLS/2014.1/bin/unwrapped/win64.o/vivado_hls.exe'
            for user 'Ag' on host 'arjun' (Windows NT_amd64 version 6.2) on
Wed Mar 04 14:11:15 -0500 2015
            in directory 'C:/Users/Ag/Documents/HLS/Classifier_layer'
@I [HLS-10] Opening project
'C:/Users/Ag/Documents/HLS/Classifier_layer/classifier'.
@I [HLS-10] Opening solution
'C:/Users/Ag/Documents/HLS/Classifier_layer/classifier/axi_wo_ar_part'.
@I [SYN-201] Setting up clock 'default' with a period of 10ns.
@I [HLS-10] Setting target device to 'xc7z020clg484-1'
   Compiling(apcc) ../../../../classifier.c in debug mode
@I [LIC-101] Checked out feature [VIVADO_HLS]
@I [HLS-10] Running
'c:/Xilinx/Vivado_HLS/2014.1/bin/unwrapped/win64.o/apcc.exe'
            for user 'Ag' on host 'arjun' (Windows NT_amd64 version 6.2) on
Wed Mar 04 14:11:39 -0500 2015
            in directory
'C:/Users/Ag/Documents/HLS/Classifier_layer/classifier/axi_wo_ar_part/csim/bu
ild'
clang: warning: c:/Xilinx/Vivado_HLS/2014.1/bin/unwrapped/win64.o/apcc.exe:
'linker' input unused when '-c' is present
clang: warning: c:/Xilinx/Vivado_HLS/2014.1/bin/unwrapped/win64.o/apcc.exe:
'linker' input unused when '-c' is present
gcc.exe: warning: c:/Xilinx/Vivado_HLS/2014.1/bin/unwrapped/win64.o/apcc.exe:
linker input file unused because linking not done
@I [APCC-3] Tmp directory is apcc_db
@I [APCC-1] APCC is done.
@I [LIC-101] Checked in feature [VIVADO_HLS]
   Generating csim.exe
*********************************************************
TEST PASSED !!!!!! The output matches the desired output.
*********************************************************
@I [SIM-1] CSim done with 0 errors.
@I [LIC-101] Checked in feature [VIVADO_HLS]
```

IV. Architectural exploration with goals

For this particular module 4 different solutions were created, each with their own set of directives:

A). No Directives:

Here the module was generated without adding any external directives.

B). Loop Directives:

Here the outer for loop was pipelined and the inner for loop was unrolled. The module being a simpler one in terms of complexity does not account for a large number of circuit elements so the idea was to let the size of the circuit increase (as it is relatively small already) and try and achieve maximum throughput and reduce latency.

C). I/O directives and array partition:

In addition to the directives for the loop, here the input and output arrays were completely partitioned and assigned AXI Lite interfaces. This attempt was made to further improve the throughput.

D). I/O directives without array partitioning

This design model does not partition arrays but assigns them with the resource directive of the type 1 port BRAM (RAM_1P_BRAM). Also the arrays were assigned AXI lite interfaces. The goal was to compare the difference in performance and resource utilization versus partitioning of the input/output arrays.

The comparative analysis for the performance and resource utilization for all the four solutions generated from Vivado HLS is as follows:

Vivado HLS Report Comparison

All Compared Solutions

axi_array_part: xc7z020clg484-1
Loop_directives: xc7z020clg484-1

axi_wo_ar_part: xc7z020clg484-1

solution_no_direc: xc7z020clg484-1

Performance Estimates

Timing (ns)

Clock		axi_array_part	Loop_directives	axi_wo_ar_part	solution_no_direc
default	Target	10.00	10.00	10.00	10.00
	Estimated	32.88	8.23	8.23	8.23

Latency (clock cycles)

•		axi_array_part	Loop_directives	axi_wo_ar_part	solution_no_direc
Latency	min	47	55	67	341
	max	47	55	67	341
Interval	min	48	56	68	342
	max	48	56	68	342

Utilization Estimates

	axi_array_part	Loop_directives	axi_wo_ar_part	solution_no_direc
BRAM_18K	0	0	0	0
DSP48E	28	13	8	8
FF	3733	1849	1529	1274
LUT	9451	3523	2708	2593

V. Finalised design for exporting RTL

- The **solution_no_direc** solution does not contain any directives and was just created to see the initial resource utilization of the module.
- The **Loop_directives** solution was created to see the effective increase in the performance by pipelining and unrolling loops. This solution still does not have interfaces directives added so resource utilisation is not to be considered the final one.
- The **axi_array_part** solution has AXI lite interfaces assigned to the I/O ports and the arrays were completely partitioned to achieve more throughput. But on synthesising this solution, the estimated clock cycle was exceedingly large and hence the performance improvement was not achieved.
- The **axi_wo_ar_part** solution also has AXI lite interface for the I/O ports but instead of partitioning the arrays they were kept as single port BRAMS. This lowers resource utilisation as compared to the previous solution. As it can be seen from the numbers present in the comparison report above, this design solution provides good performance figures in terms of clock cycles and cycle timing taken together and with relatively less resource utilization. Hence this design solution was exported as an RTL in the Vivado IP catalog.
- Creating and analysing all these solutions took less than 1 day which includes the time required for setting them correctly to get all simulation, synthesis results using the HLS tool.

VI. Exported RTL

The evaluation of the exported RTL generates the following report:

General Information

Report date: Wed Mar 04 15:20:09 -0500 2015
Device target: xc7z020clg484-1
Implementation tool: Xilinx Vivado v.2014.1

Resource Usage

	Verilog
SLICE	576
LUT	1451
FF	1415
DSP	8
BRAM	0
SRL	35

Final Timing

	Verilog
CP required	10.000
CP achieved	9.182

Timing met

The clock timing in the exported RTL is 9.182ns while that in the synthesised RTL was 8.23ns so it is comparable.

VII. C/RTL co-simulation

C/RTL co-simulation was run on the IP before exporting it as RTL in the IP catalog.

A snapshot of success of the C/RTL simulation is as follows:

****

TEST PASSED !!!!!! The output matches the desired output.

****

@I [SIM-1000] *** C/RTL co-simulation finished: PASS ***

Result:

		Latency			Interval		
RTL	Status	min	avg	max	min	avg	max
VHDL	NA	NA	NA	NA	NA	NA	NA
Verilog	Pass	80	80	80	80	80	80
SystemC	NA	NA	NA	NA	NA	NA	NA

VIII. Packaged IP in Vivado

The packaged IP exported to Vivado seems like this:

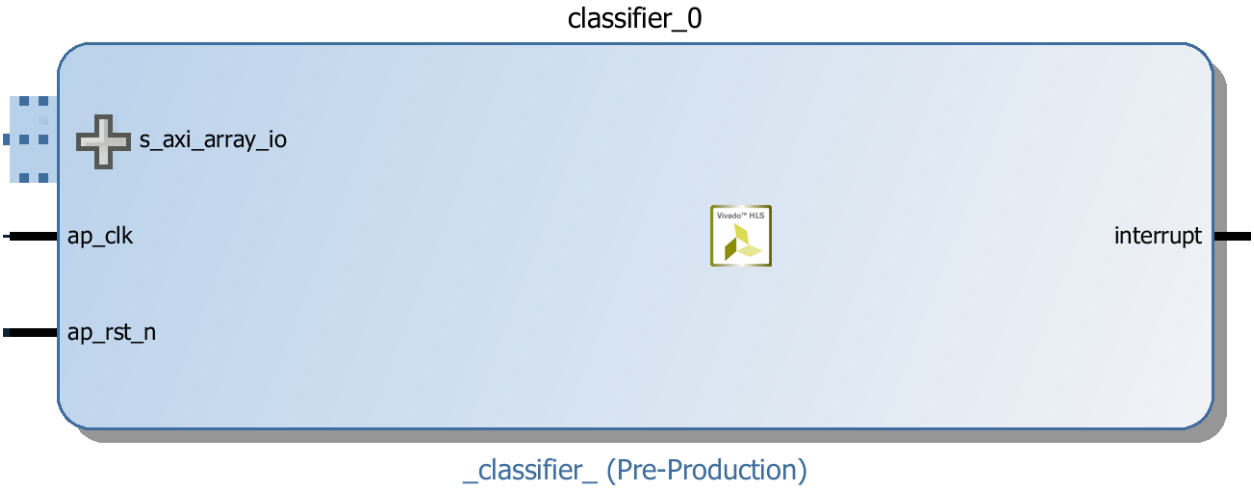

Figure 2 Exported classifier module

IX. Microblaze system

Possible interaction of classifier and microblaze

One way to test a standalone module of classifier using a microblaze controller would be to provide certain types of input vectors whose classification is already known. Then wait for the classifier layer to generate the output for those test vectors, collect the output data and compare it with the already known (reference) output. If the outputs match the reference, the test passes and that can be indicated by glowing a led.

The images below show the block diagrams for the classifier module generated by Vivado HLS integrated with the microblaze controller in Vivado along with the necessary components.

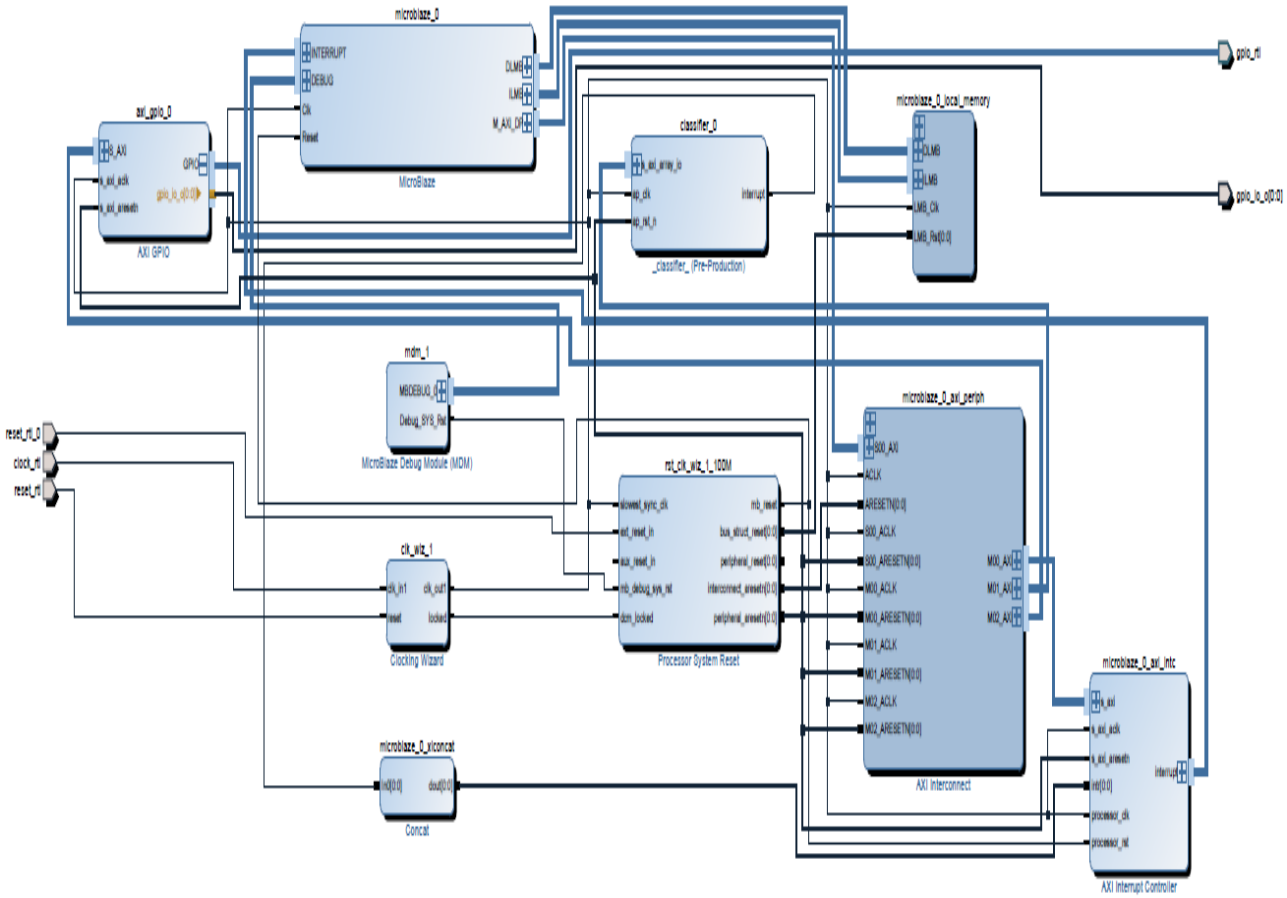

Figure 3 Block diagram of the entire system

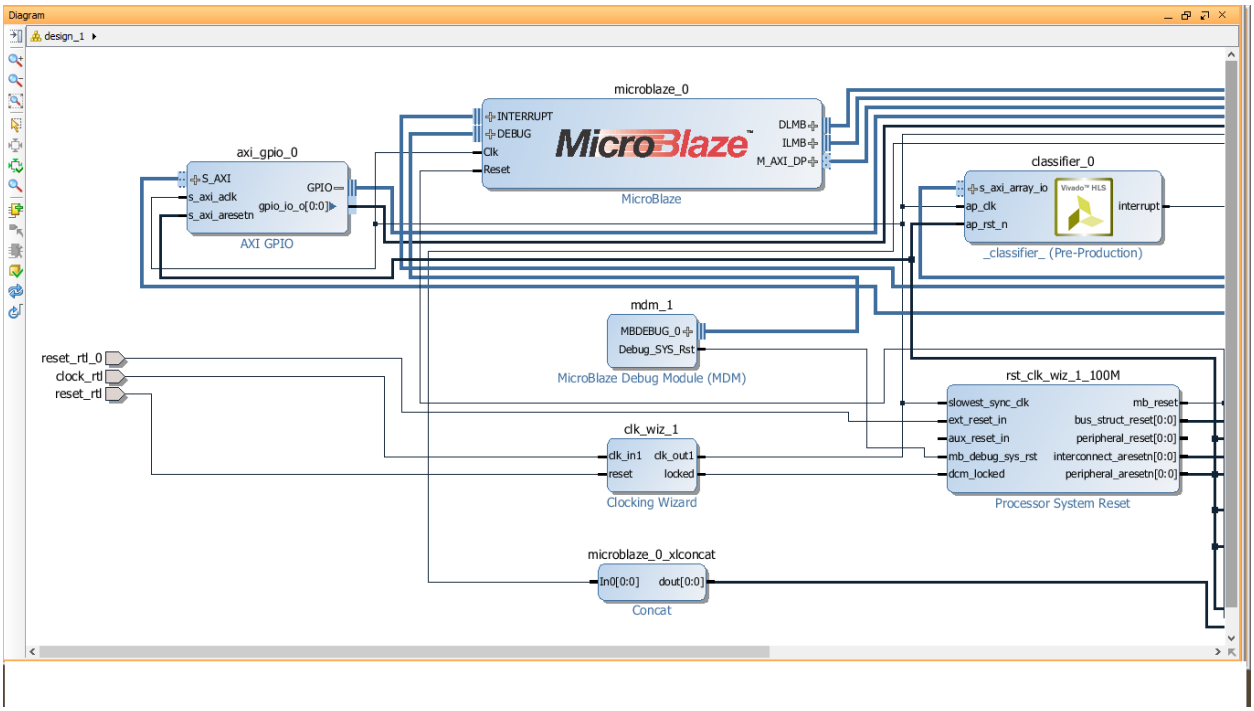

Figure 4 Zoomed view of the circuit (Left side)

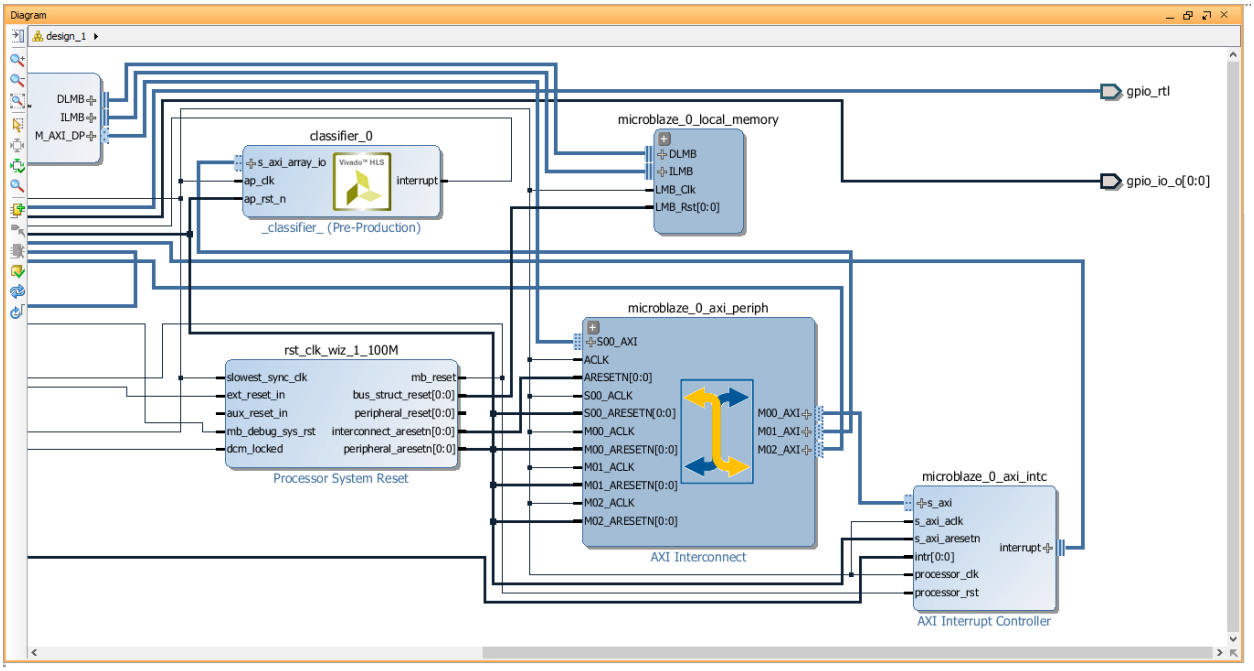

Figure 5 Zoomed view of the circuit (Right side)

The following image shows the synthesised circuit:

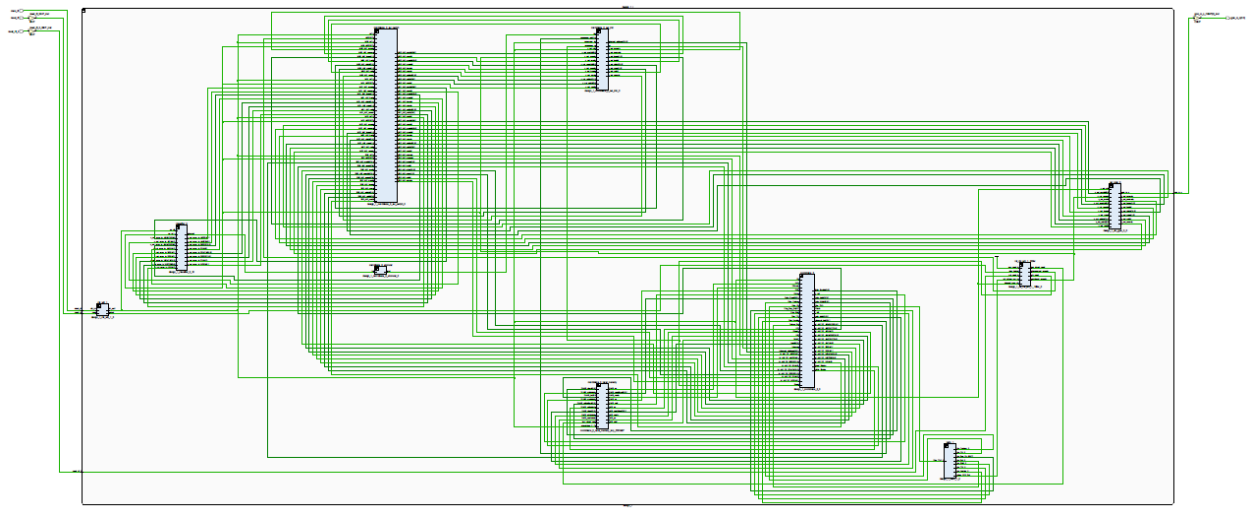

Figure 6 Synthesised circuit

After running the implementation, the implemented circuit seems like this:

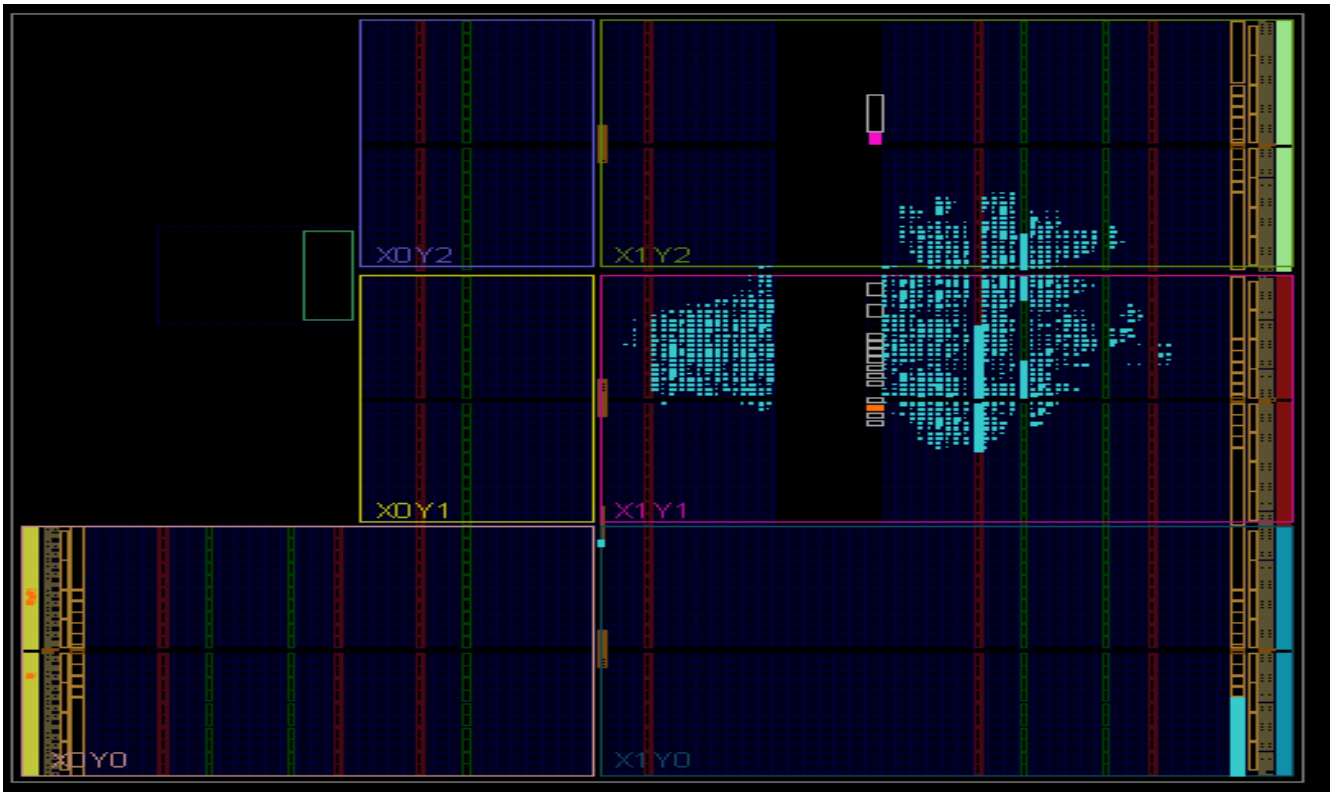

Figure 7 Implemented design

The log for successful bit stream generation is as follows:

#---

Vivado v2014.1 (64-bit)

SW Build 881834 on Fri Apr 4 14:15:54 MDT 2014

IP Build 877625 on Fri Mar 28 16:29:15 MDT 2014

Start of session at: Wed Mar 04 20:59:19 2015

Process ID: 6948

Log file: C:/Users/Ag/tue_afternoon/tue_afternoon.runs/impl_1/design_1_wrapper.vdi

Journal file: C:/Users/Ag/tue_afternoon/tue_afternoon.runs/impl_1\vivado.jou

#---

source design_1_wrapper.tcl -notrace

Command: open_checkpoint design_1_wrapper_routed.dcp

INFO: [Netlist 29-17] Analyzing 144 Unisim elements for replacement

INFO: [Netlist 29-28] Unisim Transformation completed in 0 CPU seconds

INFO: [Project 1-479] Netlist was created with Vivado 2014.1

Loading clock regions from
C:/Xilinx/Vivado/2014.1/data\parts/xilinx/zynq/zynq/xc7z020/ClockRegion.xml

Loading clock buffers from
C:/Xilinx/Vivado/2014.1/data\parts/xilinx/zynq/zynq/xc7z020/ClockBuffers.xml

Loading clock placement rules from
C:/Xilinx/Vivado/2014.1/data/parts/xilinx/zynq/ClockPlacerRules.xml

Loading package pin functions from
C:/Xilinx/Vivado/2014.1/data\parts/xilinx/zynq/PinFunctions.xml...

Loading package from
C:/Xilinx/Vivado/2014.1/data\parts/xilinx/zynq/zynq/xc7z020/clg484/Package.xml

Loading io standards from C:/Xilinx/Vivado/2014.1/data\./parts/xilinx/zynq/IOStandards.xml

Parsing XDC File [C:/Users/Ag/tue_afternoon/tue_afternoon.runs/impl_1/.Xil/Vivado-6948-Arjun/dcp/design_1_wrapper_early.xdc]

INFO: [Timing 38-35] Done setting XDC timing constraints.
[C:/Users/Ag/tue_afternoon/tue_afternoon.srcs/sources_1/bd/design_1/ip/design_1_mdm_1_0/design_1_mdm_1_0.xdc:50]

get_clocks: Time (s): cpu = 00:00:21 ; elapsed = 00:00:25 . Memory (MB): peak = 902.352 ; gain = 416.871

INFO: [Timing 38-2] Deriving generated clocks [C:/Users/Ag/tue_afternoon/tue_afternoon.srcs/sources_1/bd/design_1/ip/design_1_clk_wiz_1_0/design_1_clk_wiz_1_0.xdc:56]

Finished Parsing XDC File [C:/Users/Ag/tue_afternoon/tue_afternoon.runs/impl_1/.Xil/Vivado-6948-Arjun/dcp/design_1_wrapper_early.xdc]

Parsing XDC File [C:/Users/Ag/tue_afternoon/tue_afternoon.runs/impl_1/.Xil/Vivado-6948-Arjun/dcp/design_1_wrapper.xdc]

Finished Parsing XDC File [C:/Users/Ag/tue_afternoon/tue_afternoon.runs/impl_1/.Xil/Vivado-6948-Arjun/dcp/design_1_wrapper.xdc]

Parsing XDC File [C:/Users/Ag/tue_afternoon/tue_afternoon.runs/impl_1/.Xil/Vivado-6948-Arjun/dcp/design_1_wrapper_late.xdc]

Finished Parsing XDC File [C:/Users/Ag/tue_afternoon/tue_afternoon.runs/impl_1/.Xil/Vivado-6948-Arjun/dcp/design_1_wrapper_late.xdc]

Reading XDEF placement.

Reading XDEF routing.

Read XDEF File: Time (s): cpu = 00:00:01 ; elapsed = 00:00:01 . Memory (MB): peak = 904.895 ; gain = 2.543

Restoring placement.

Restored 1145 out of 1145 XDEF sites from archive | CPU: 2.000000 secs | Memory: 0.000000 MB |

INFO: [Opt 31-138] Pushed 0 inverter(s).

INFO: [Project 1-111] Unisim Transformation Summary:

A total of 127 instances were transformed.

LUT6_2 => LUT6_2 (LUT5, LUT6): 79 instances

RAM16X1D => RAM32X1D (RAMD32, RAMD32): 32 instances

RAM32M => RAM32M (RAMD32, RAMD32, RAMD32, RAMD32, RAMD32, RAMD32, RAMS32, RAMS32): 16 instances

INFO: [Project 1-484] Checkpoint was created with build 881834

open_checkpoint: Time (s): cpu = 00:00:34 ; elapsed = 00:01:03 . Memory (MB): peak = 909.813 ; gain = 737.422

Attempting to get a license for feature 'Implementation' and/or device 'xc7z020'

INFO: [Common 17-349] Got license for feature 'Implementation' and/or device 'xc7z020'

Running DRC as a precondition to command write_bitstream

INFO: [Drc 23-27] Running DRC with 2 threads

INFO: [Vivado 12-3199] DRC finished with 0 Errors, 18 Warnings, 8 Advisories

INFO: [Vivado 12-3200] Please refer to the DRC report (report_drc) for more information.

Loading data files...

Loading site data...

Loading route data...

Processing options...

Creating bitmap...

Creating bitstream...

Writing bitstream ./design_1_wrapper.bit...

INFO: [Vivado 12-1842] Bitgen Completed Successfully.

INFO: [Project 1-118] WebTalk data collection is enabled (User setting is ON. Install Setting is ON.).

INFO: [Common 17-186] 'C:/Users/Ag/tue_afternoon/tue_afternoon.runs/impl_1/usage_statistics_webtalk.xml' has been successfully sent to Xilinx on Wed Mar 04 21:02:40 2015. For additional details about this file, please refer to the WebTalk help file at C:/Xilinx/Vivado/2014.1/doc/webtalk_introduction.html.

INFO: [Common 17-83] Releasing license: Implementation

write_bitstream: Time (s): cpu = 00:00:50 ; elapsed = 00:02:13 . Memory (MB): peak = 1270.977 ; gain = 361.164

INFO: [Common 17-206] Exiting Vivado at Wed Mar 04 21:02:40 2015...