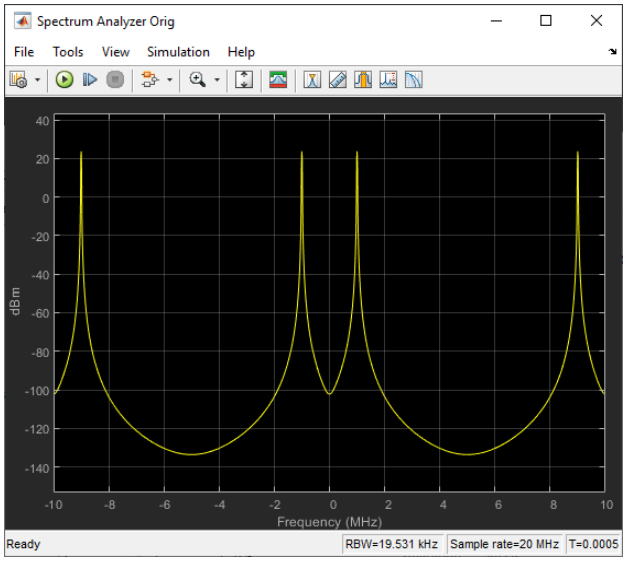
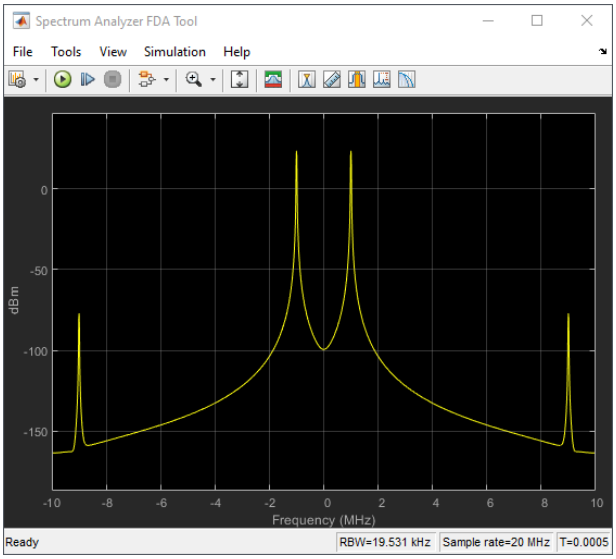
ELEC 522 Project 1
Aedan Cullen

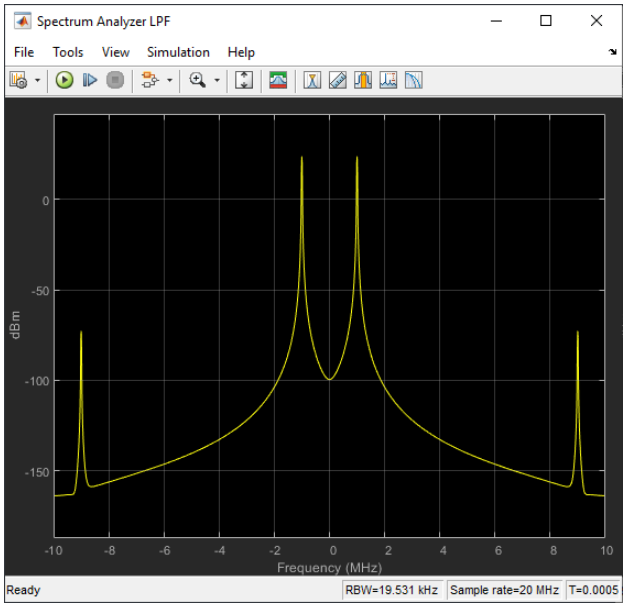
Lab 1

Step 1

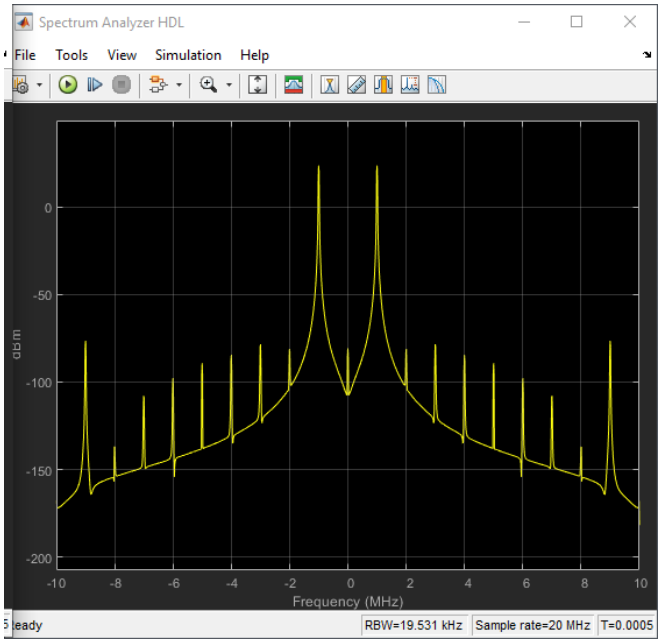
a. Screen capture of spectrum plots of the initial waveforms



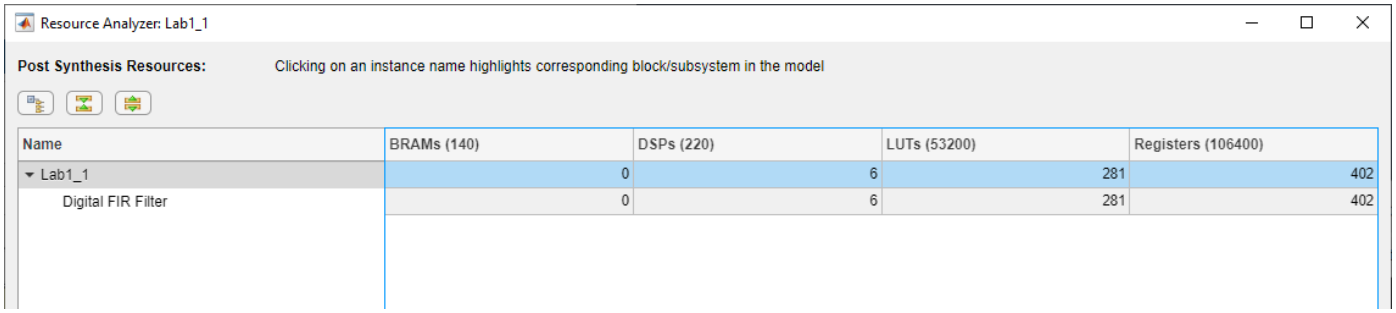




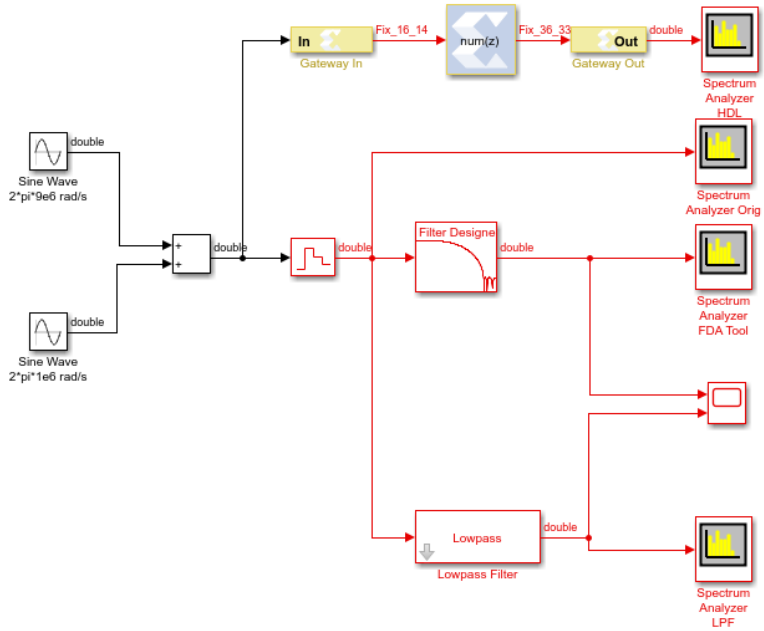
b. Screen capture of spectrum plots after adding Digital FIR Filter block



c. Screen capture of resource utilization output



d. Screen capture of final block diagram



e. Compare your results with the tutorial results and note any differences or if the same

No differences were noticed. As expected, 6 DSP blocks are used. The resource utilization matches the tutorial exactly.

Step 2

a. Screen capture of resource utilization output for higher frequency in Generate step

Resource Analyzer: Lab1_2

Post Synthesis Resources: Clicking on an instance name highlights corresponding block/subsystem in the model

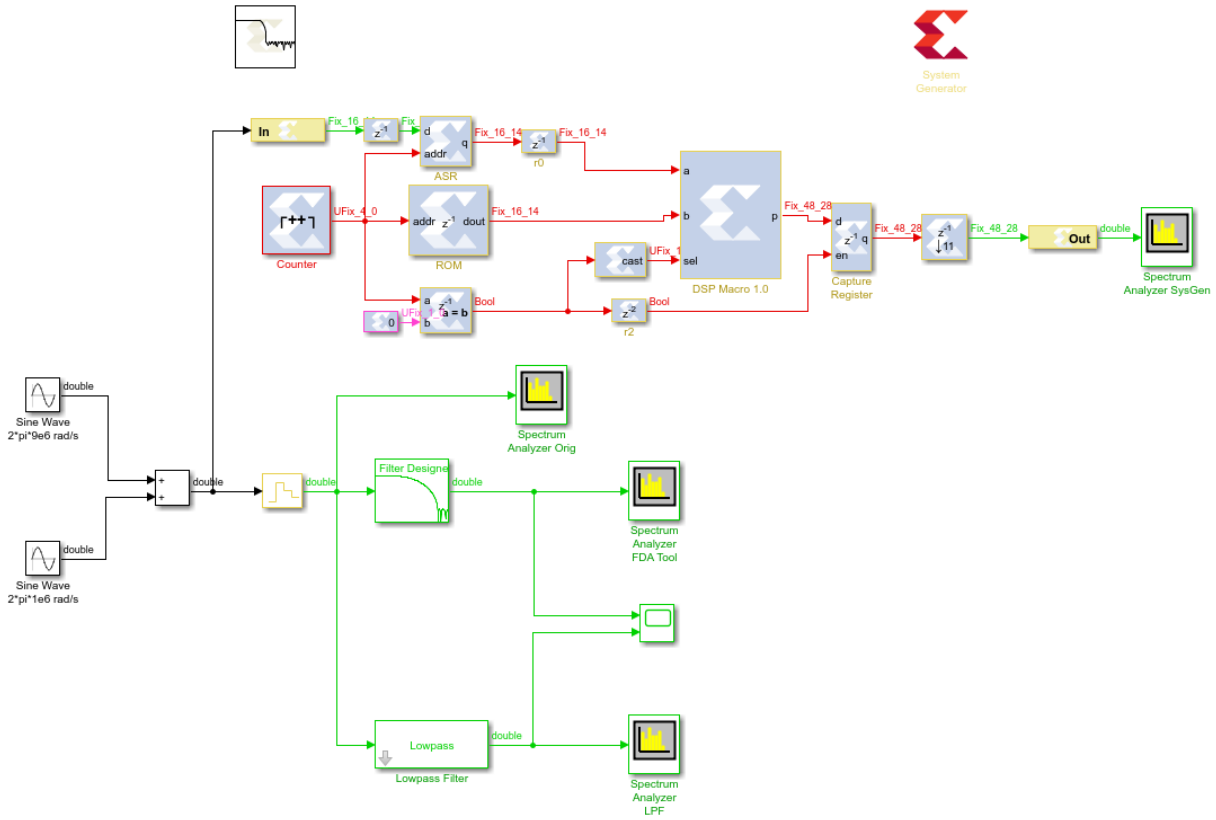
Name	BRAMs (140)	DSPs (220)	LUTs (53200)	Registers (106400)
Lab1_2	0	1	105	196
Digital FIR Filter	0	1	105	196

b. Compare your results with the tutorial results and note any differences or if the same.

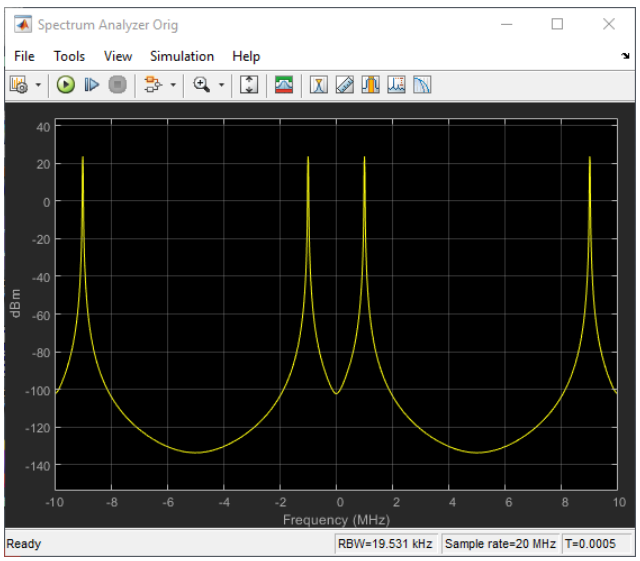
No differences were noticed. As expected, only 1 DSP block is now used to accomplish the same task, due to the higher frequency.

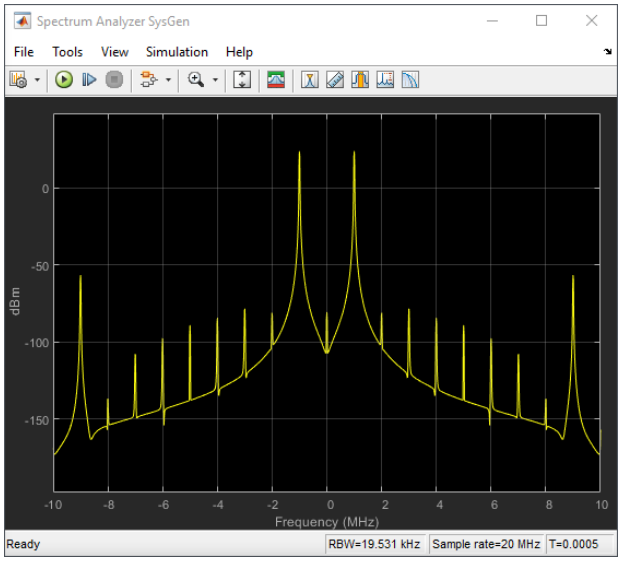
Step 3

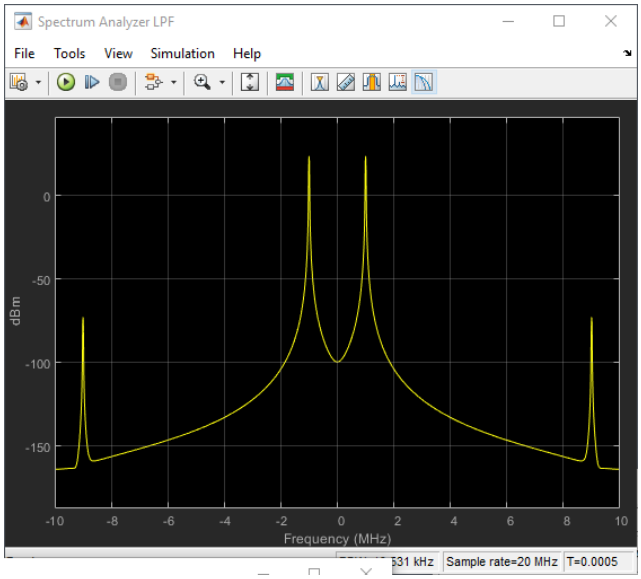
a. Screen capture of final block diagram using discrete components

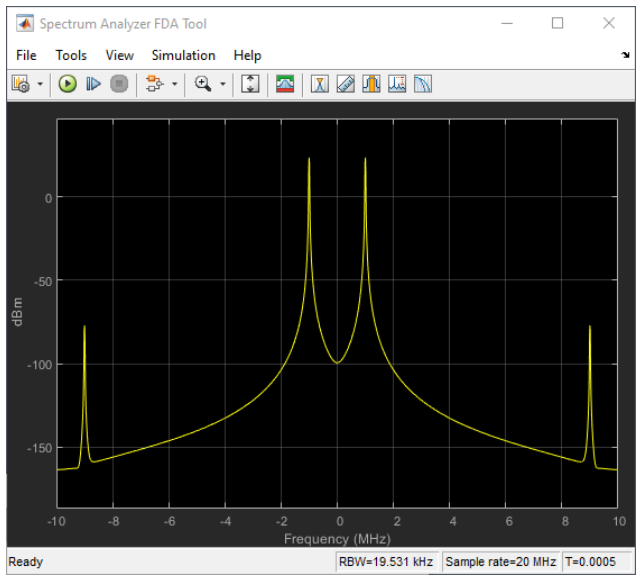


b. Screen capture of spectrum plots

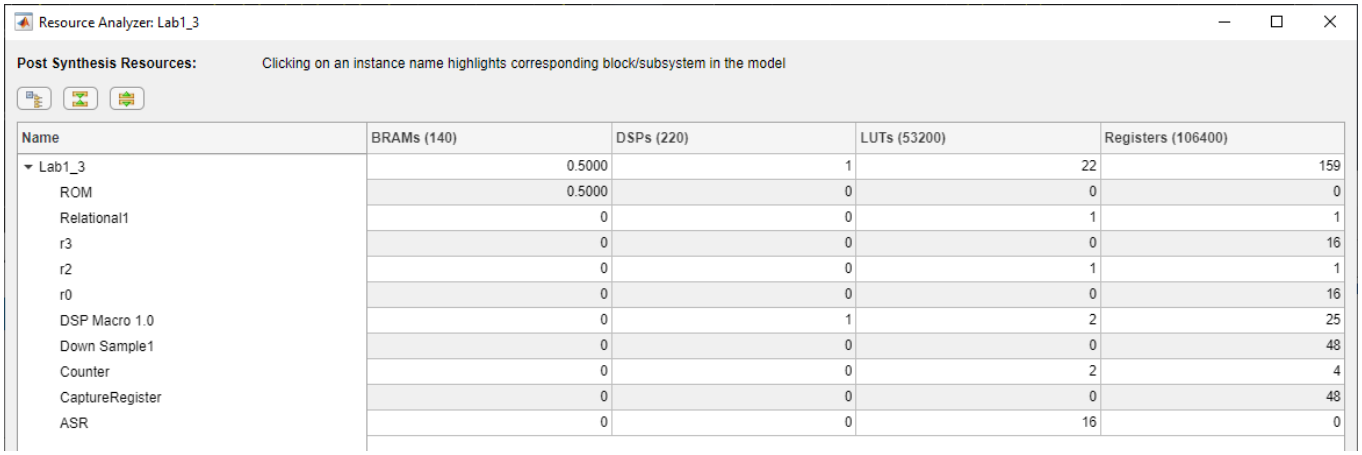








c. Screen capture of resource utilization output



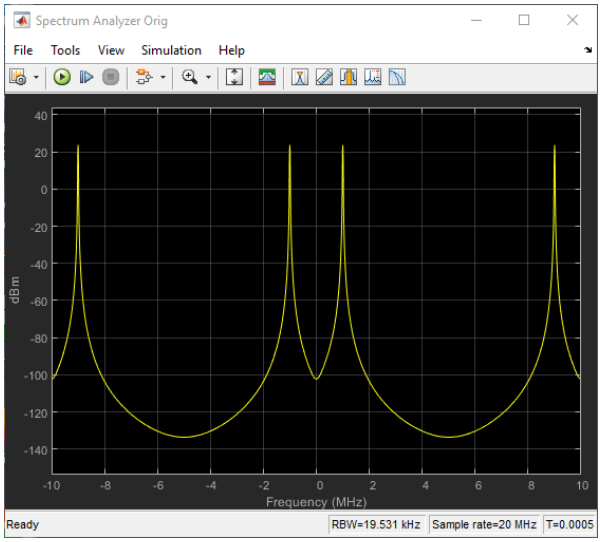
Name	BRAMs (140)	DSPs (220)	LUTs (53200)	Registers (106400)
Lab1_3	0.5000	1	22	159
ROM	0.5000	0	0	0
Relational1	0	0	1	1
r3	0	0	0	16
r2	0	0	1	1
r0	0	0	0	16
DSP Macro 1.0	0	1	2	25
Down Sample1	0	0	0	48
Counter	0	0	2	4
CaptureRegister	0	0	0	48
ASR	0	0	16	0

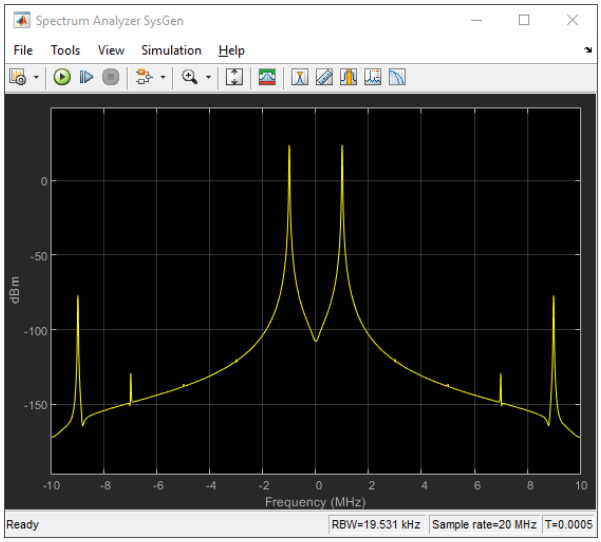
d. Compare your results with the tutorial results and note any differences or if the same

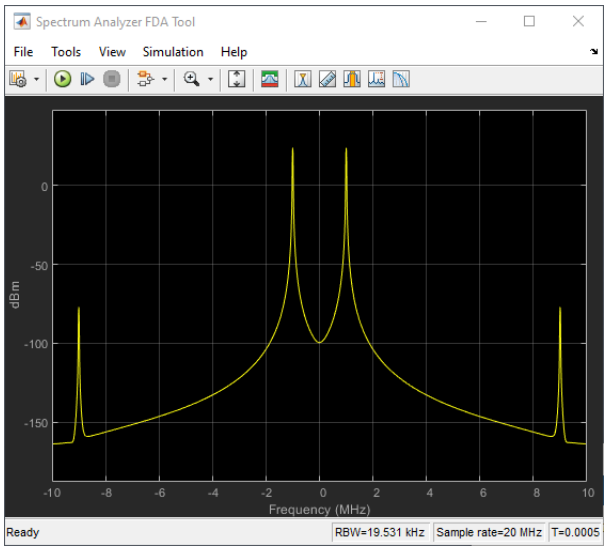
No differences were noticed. The resource utilization matches the tutorial exactly.

Step 4

a. Part 1, Screen capture of spectrum plots of floating-point version


Spectrum Analyzer Orig
File Tools View Simulation Help
dBm
Frequency (MHz)
Ready
RBW=19.531 kHz
Sample rate=20 MHz
T=0.0005


Spectrum Analyzer SysGen
File Tools View Simulation Help
dBm
Frequency (MHz)
Ready
RBW=19.531 kHz
Sample rate=20 MHz
T=0.0005


Spectrum Analyzer FDA Tool
File Tools View Simulation Help
dBm
Frequency (MHz)
Ready
RBW=19.531 kHz
Sample rate=20 MHz
T=0.0005

b. Part 1, Screen capture of floating-point resource utilization output

Resource Analyzer: Lab1_4_1

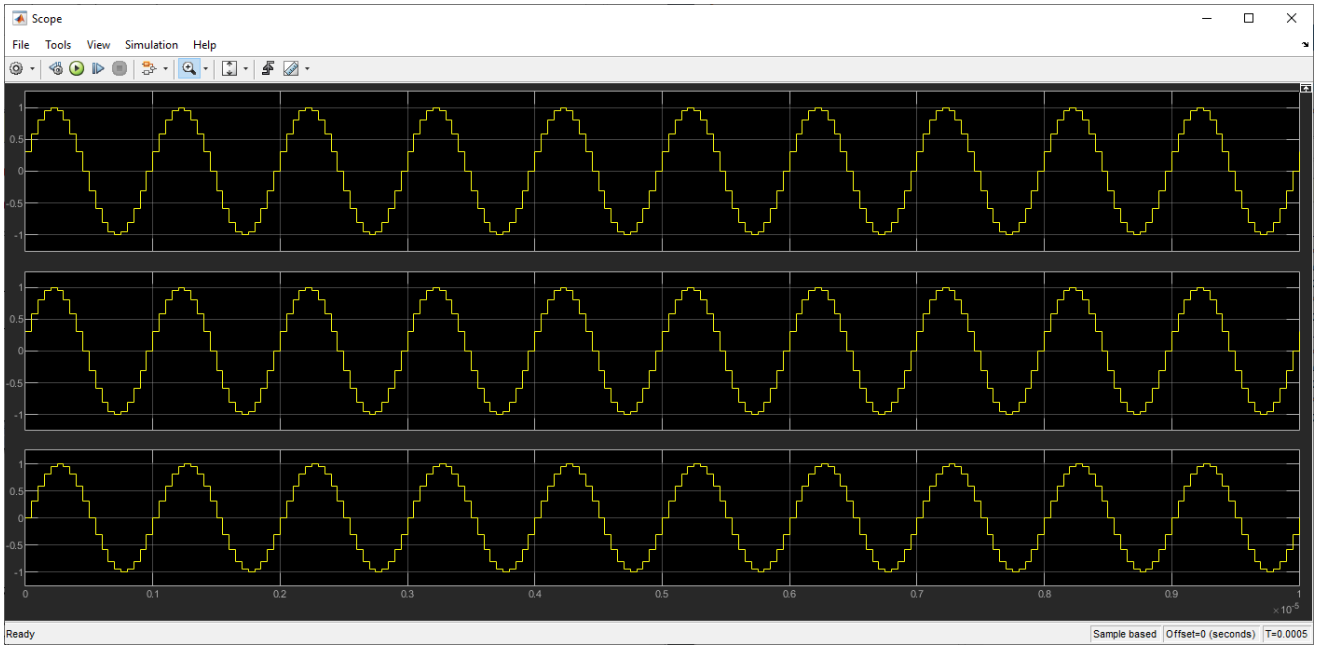
Post Synthesis Resources: Clicking on an instance name highlights corresponding block/subsystem in the model

Name	BRAMs (140)	DSPs (220)	LUTs (53200)	Registers (106400)
Lab1_4_1	0	33	5145	1299
FIR	0	33	5145	1299
Mult9	0	3	68	56
Mult8	0	3	67	56
Mult7	0	3	68	56
Mult6	0	3	68	56
Mult5	0	3	68	56
Mult4	0	3	67	56
Mult3	0	3	68	56
Mult2	0	3	67	56
Mult11	0	3	67	56
Mult10	0	3	67	56
Mult1	0	3	67	56
Delay9	0	0	32	32
Delay8	0	0	32	32
Delay7	0	0	32	32
Delay6	0	0	32	32
Delay5	0	0	32	32
Delay4	0	0	32	32
Delay3	0	0	32	32
Delay2	0	0	32	32
Delay10	0	0	32	32
Delay1	0	0	32	32
AddSub9	0	0	379	33
AddSub8	0	0	379	33
AddSub7	0	0	379	33
AddSub6	0	0	379	33
AddSub5	0	0	379	33
AddSub4	0	0	379	33
AddSub3	0	0	379	33
AddSub2	0	0	379	33
AddSub11	0	0	379	33
AddSub10	0	0	379	33
AddSub1	0	0	293	33

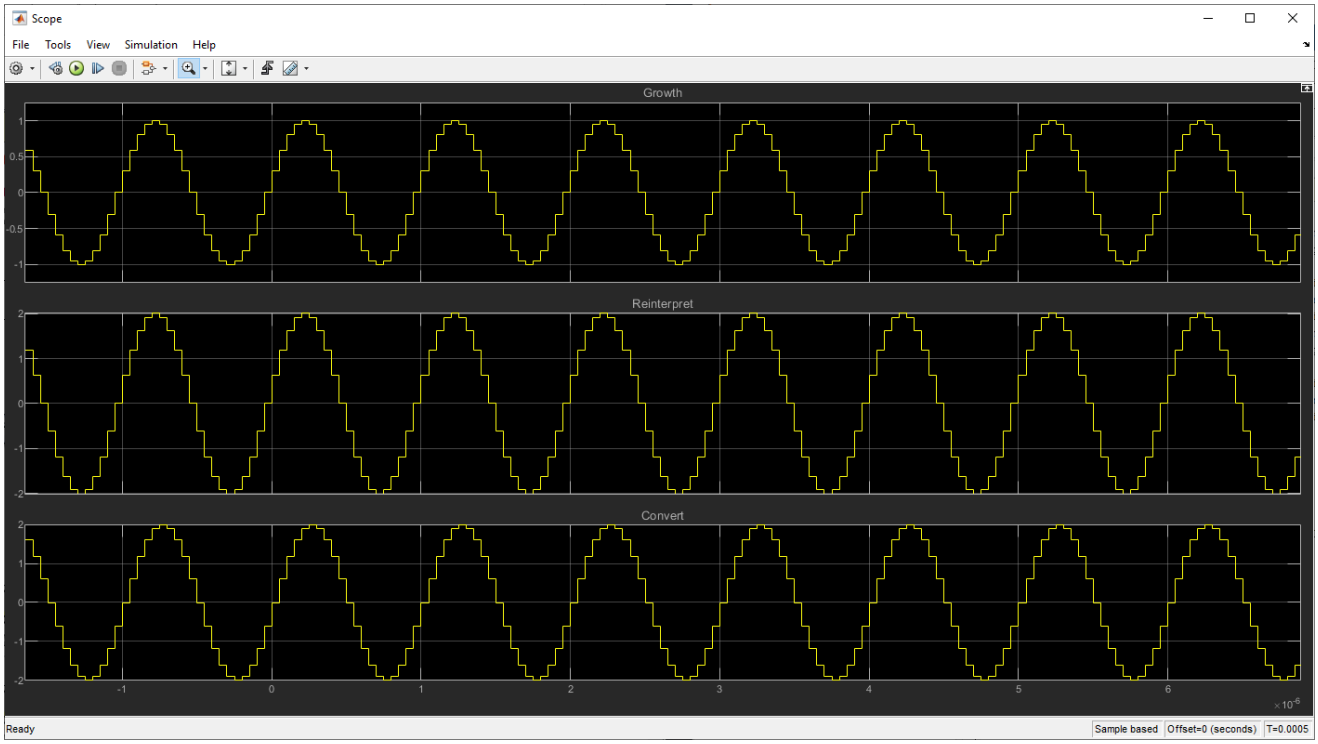
c. Part 1, Compare your results with the tutorial results and note any differences or if the same

No differences were noticed. The resource utilization matches the tutorial exactly.

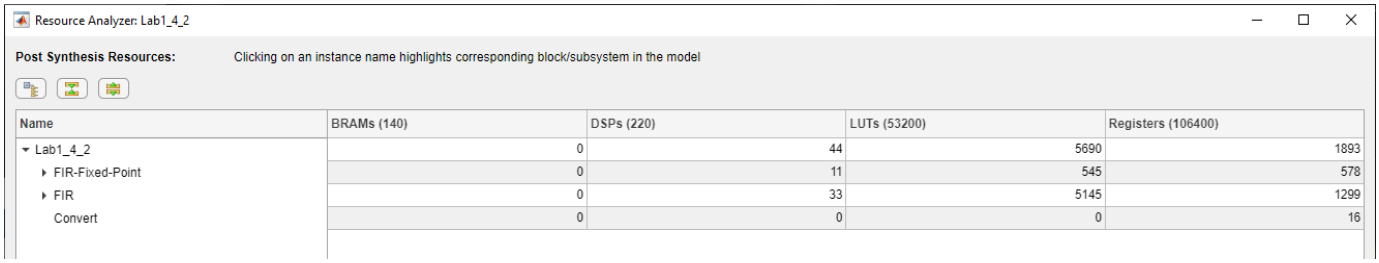
d. Part 2, Screen capture of scope results for fixed-point version – initial version



e. Part 2, Screen capture of scope results for fixed-point version – after changing reinterpret and convert blocks



f. Part 2, Screen capture of floating-point resource utilization output



Name	BRAMs (140)	DSPs (220)	LUTs (53200)	Registers (106400)
Lab1_4_2	0	44	5690	1893
FIR-Fixed-Point	0	11	545	578
FIR	0	33	5145	1299
Convert	0	0	0	16

g. Part 2, Compare your results with the tutorial results and note any differences or if the same

No differences were noticed. The resource utilization matches the tutorial exactly.

Lab 2

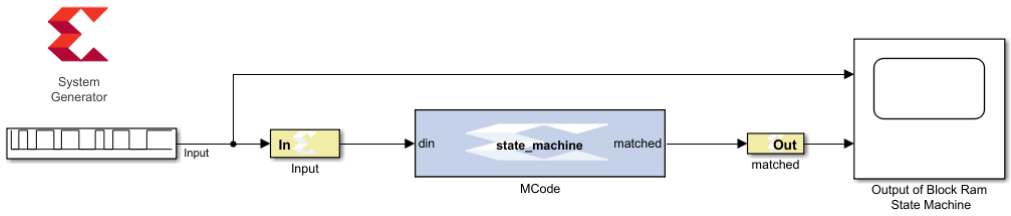
Step 1

a. Include your state_machine.m code

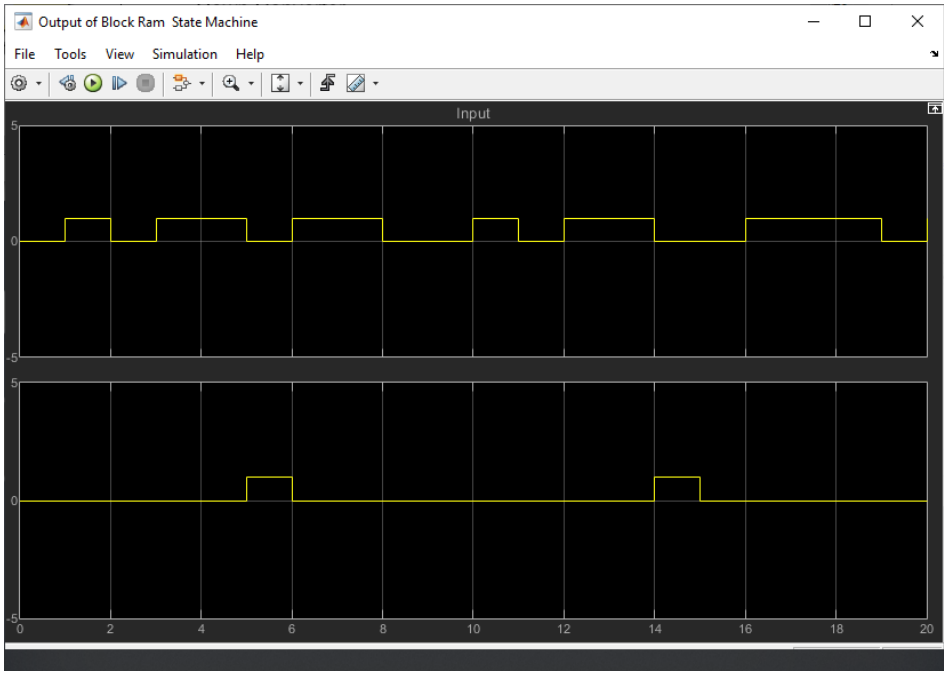
A screenshot is provided below, and the file is attached to Canvas along with this submission.

```
function matched = state_machine(din)
    persistent state, state = xl_state(0, {xlUnsigned, 3, 0});
    matched = 0;
    switch state
        case 0
            if din == 1
                state = 1;
            else
                state = 0;
            end
        case 1
            if din == 1
                state = 1;
            else
                state = 2;
            end
        case 2
            if din == 1
                state = 3;
            else
                state = 0;
            end
        case 3
            if din == 1
                state = 4;
            else
                state = 2;
            end
        case 4
            if din == 1
                state = 1;
            else
                state = 0;
            matched = 1;
            end
        otherwise
    end
end
```

b. Screen capture of final block diagram



c. Screen capture of output waveform

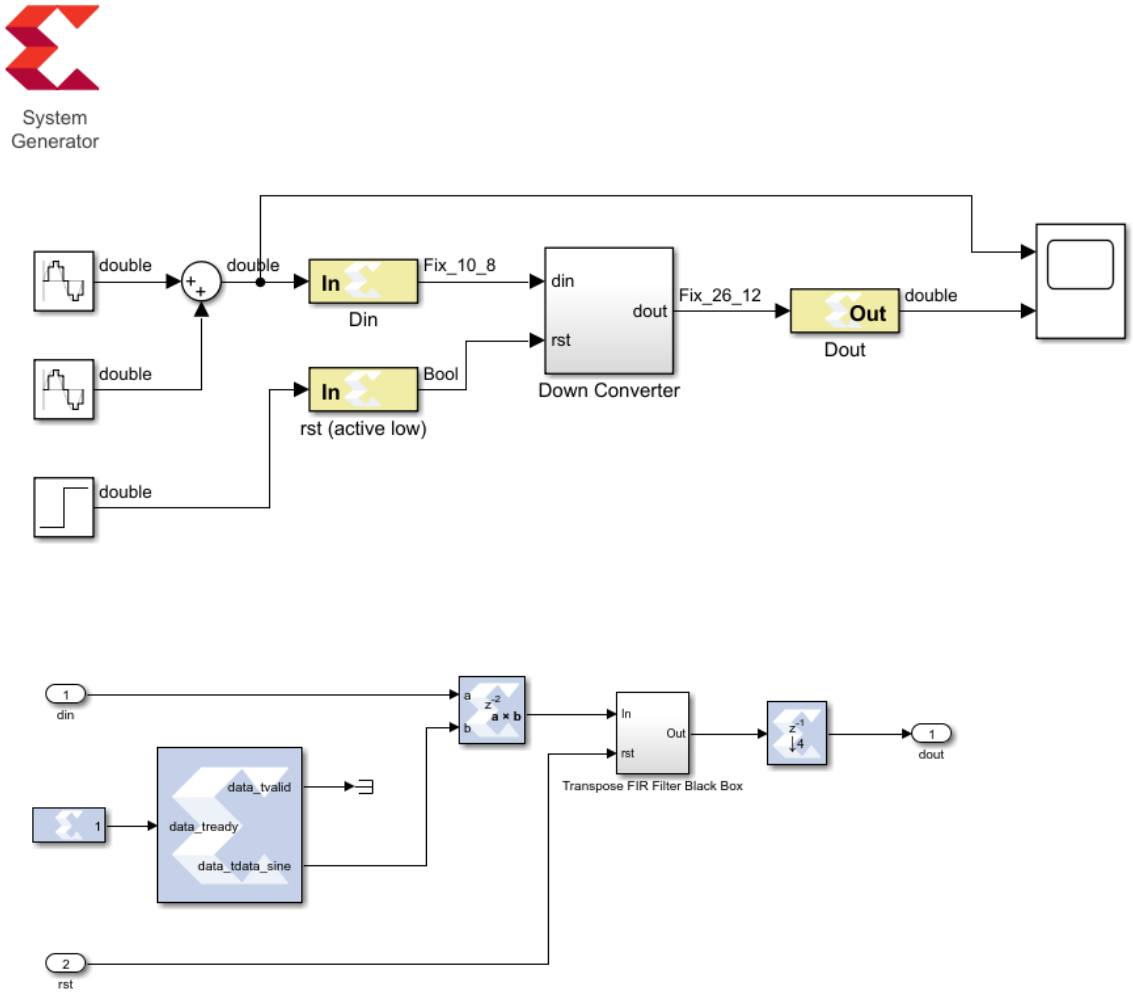


d. Compare your results with the tutorial results and note any differences or if the same

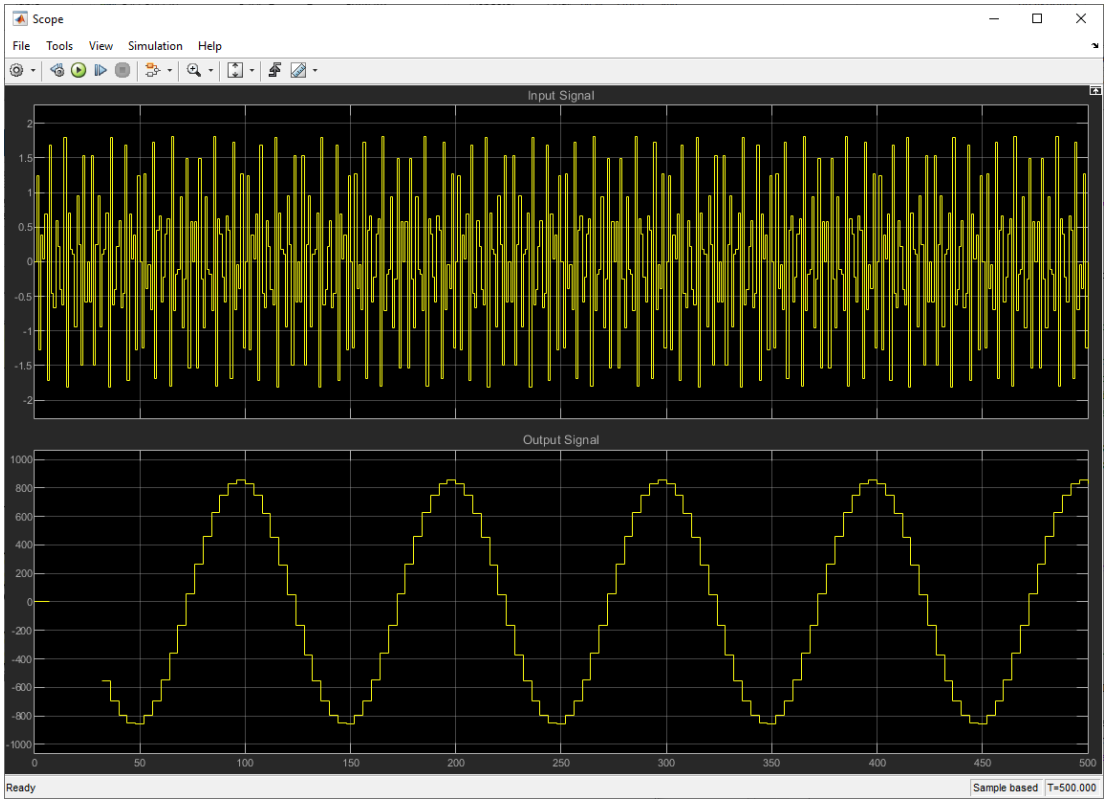
No differences were noticed. The output waveform matches that of the tutorial.

Step 2

a. Screen capture of final block diagram



b. Screen capture of final output waveform



c. Compare your results with the tutorial results and note any differences or if the same

No differences were noticed. The scope waveforms match those of the tutorial.

Lab 3

Step 1

a. Screen capture of timing analyzer results – realize that the Kintex -3 part will have less timing error than the Zynq Artix -1 part

For Zynq 7020 -1 speed grade (Artix PL):

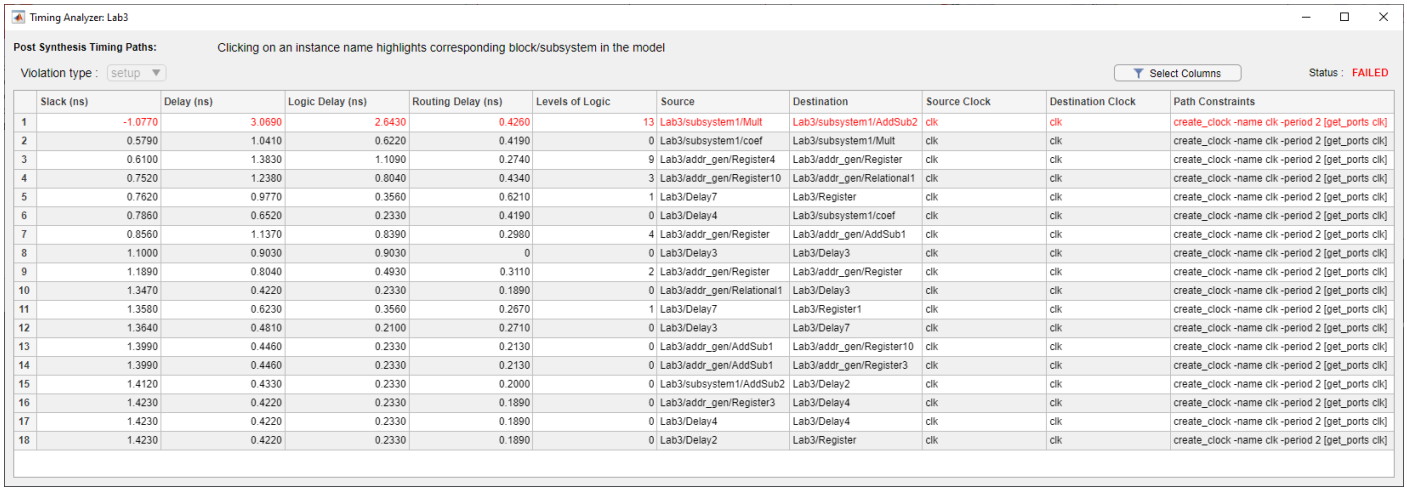
Timing Analyzer: Lab3

Post Synthesis Timing Paths: Clicking on an instance name highlights corresponding block/subsystem in the model

Violation type : setup | Select Columns | Status : FAILED

	Slack (ns)	Delay (ns)	Logic Delay (ns)	Routing Delay (ns)	Levels of Logic	Source	Destination	Source Clock	Destination Clock	Path Constraints
1	-3.2650	5.2900	4.4810	0.8090	14	Lab3/subsystem1/Mult	Lab3/subsystem1/AddSub2	clk	clk	create_clock -name clk -period 2 [get_ports clk]
2	-0.8390	2.8640	2.3850	0.4790	9	Lab3/addr_gen/Register4	Lab3/addr_gen/Register	clk	clk	create_clock -name clk -period 2 [get_ports clk]
3	-0.4450	2.4550	1.6940	0.7610	3	Lab3/addr_gen/Register10	Lab3/addr_gen/Relational1	clk	clk	create_clock -name clk -period 2 [get_ports clk]
4	-0.2780	2.3030	1.8000	0.5030	4	Lab3/addr_gen/Register	Lab3/addr_gen/AddSub1	clk	clk	create_clock -name clk -period 2 [get_ports clk]
5	-0.2160	1.6820	0.8820	0.8000	0	Lab3/subsystem1/coef	Lab3/subsystem1/Mult	clk	clk	create_clock -name clk -period 2 [get_ports clk]
6	-0.1430	1.3180	0.5180	0.8000	0	Lab3/Delay4	Lab3/subsystem1/coef	clk	clk	create_clock -name clk -period 2 [get_ports clk]
7	-0.0910	1.8380	0.8130	1.0250	1	Lab3/Delay7	Lab3/Register	clk	clk	create_clock -name clk -period 2 [get_ports clk]
8	0.4170	1.6170	1.6170	0	0	Lab3/Delay3	Lab3/Delay3	clk	clk	create_clock -name clk -period 2 [get_ports clk]
9	0.4480	1.5770	1.0690	0.5080	2	Lab3/addr_gen/Register	Lab3/addr_gen/Register	clk	clk	create_clock -name clk -period 2 [get_ports clk]
10	0.7100	1.2830	0.8130	0.4700	1	Lab3/Delay7	Lab3/Register1	clk	clk	create_clock -name clk -period 2 [get_ports clk]
11	0.8350	0.8940	0.4780	0.4160	0	Lab3/Delay3	Lab3/Delay7	clk	clk	create_clock -name clk -period 2 [get_ports clk]
12	0.8490	0.8520	0.5180	0.3340	0	Lab3/addr_gen/Relational1	Lab3/Delay3	clk	clk	create_clock -name clk -period 2 [get_ports clk]
13	0.8530	0.8760	0.5180	0.3580	0	Lab3/addr_gen/AddSub1	Lab3/addr_gen/Register10	clk	clk	create_clock -name clk -period 2 [get_ports clk]
14	0.8530	0.8760	0.5180	0.3580	0	Lab3/addr_gen/AddSub1	Lab3/addr_gen/Register3	clk	clk	create_clock -name clk -period 2 [get_ports clk]
15	0.8660	0.8630	0.5180	0.3450	0	Lab3/subsystem1/AddSub2	Lab3/Delay2	clk	clk	create_clock -name clk -period 2 [get_ports clk]
16	0.8770	0.8520	0.5180	0.3340	0	Lab3/addr_gen/Register3	Lab3/Delay4	clk	clk	create_clock -name clk -period 2 [get_ports clk]
17	0.8770	0.8520	0.5180	0.3340	0	Lab3/Delay4	Lab3/Delay4	clk	clk	create_clock -name clk -period 2 [get_ports clk]
18	0.8770	0.8520	0.5180	0.3340	0	Lab3/Delay2	Lab3/Register	clk	clk	create_clock -name clk -period 2 [get_ports clk]

For Zynq 7045 -2 speed grade, as found on the ZC706 evaluation board (Kintex PL):


Timing Analyzer: Lab3

Post Synthesis Timing Paths: Clicking on an instance name highlights corresponding block/subsystem in the model

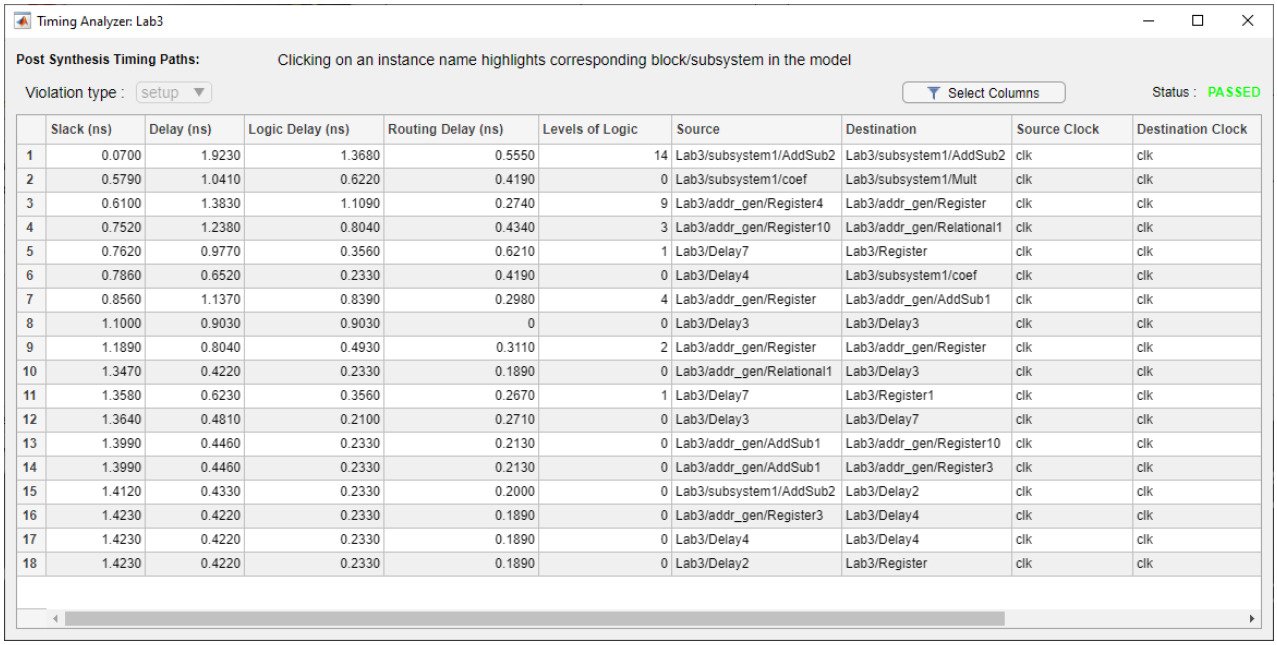
Violation type : setup — Select Columns — Status : FAILED

	Slack (ns)	Delay (ns)	Logic Delay (ns)	Routing Delay (ns)	Levels of Logic	Source	Destination	Source Clock	Destination Clock	Path Constraints
1	-1.0770	3.0690	2.6430	0.4260	13	Lab3/subsystem1/Mult	Lab3/subsystem1/AddSub2	clk	clk	create_clock -name clk -period 2 [get_ports clk]
2	0.5790	1.0410	0.6220	0.4190	0	Lab3/subsystem1/coef	Lab3/subsystem1/Mult	clk	clk	create_clock -name clk -period 2 [get_ports clk]
3	0.6100	1.3830	1.1090	0.2740	9	Lab3/addr_gen/Register4	Lab3/addr_gen/Register	clk	clk	create_clock -name clk -period 2 [get_ports clk]
4	0.7520	1.2380	0.8040	0.4340	3	Lab3/addr_gen/Register10	Lab3/addr_gen/Relational1	clk	clk	create_clock -name clk -period 2 [get_ports clk]
5	0.7620	0.9770	0.3560	0.6210	1	Lab3/Delay7	Lab3/Register	clk	clk	create_clock -name clk -period 2 [get_ports clk]
6	0.7860	0.6520	0.2330	0.4190	0	Lab3/Delay4	Lab3/subsystem1/coef	clk	clk	create_clock -name clk -period 2 [get_ports clk]
7	0.8560	1.1370	0.8390	0.2980	4	Lab3/addr_gen/Register	Lab3/addr_gen/AddSub1	clk	clk	create_clock -name clk -period 2 [get_ports clk]
8	1.1000	0.9030	0.9030	0	0	Lab3/Delay3	Lab3/Delay3	clk	clk	create_clock -name clk -period 2 [get_ports clk]
9	1.1890	0.8040	0.4930	0.3110	2	Lab3/addr_gen/Register	Lab3/addr_gen/Register	clk	clk	create_clock -name clk -period 2 [get_ports clk]
10	1.3470	0.4220	0.2330	0.1890	0	Lab3/addr_gen/Relational1	Lab3/Delay3	clk	clk	create_clock -name clk -period 2 [get_ports clk]
11	1.3580	0.6230	0.3560	0.2670	1	Lab3/Delay7	Lab3/Register1	clk	clk	create_clock -name clk -period 2 [get_ports clk]
12	1.3640	0.4810	0.2100	0.2710	0	Lab3/Delay3	Lab3/Delay7	clk	clk	create_clock -name clk -period 2 [get_ports clk]
13	1.3990	0.4460	0.2330	0.2130	0	Lab3/addr_gen/AddSub1	Lab3/addr_gen/Register10	clk	clk	create_clock -name clk -period 2 [get_ports clk]
14	1.3990	0.4460	0.2330	0.2130	0	Lab3/addr_gen/AddSub1	Lab3/addr_gen/Register3	clk	clk	create_clock -name clk -period 2 [get_ports clk]
15	1.4120	0.4330	0.2330	0.2000	0	Lab3/subsystem1/AddSub2	Lab3/Delay2	clk	clk	create_clock -name clk -period 2 [get_ports clk]
16	1.4230	0.4220	0.2330	0.1890	0	Lab3/addr_gen/Register3	Lab3/Delay4	clk	clk	create_clock -name clk -period 2 [get_ports clk]
17	1.4230	0.4220	0.2330	0.1890	0	Lab3/Delay4	Lab3/Delay4	clk	clk	create_clock -name clk -period 2 [get_ports clk]
18	1.4230	0.4220	0.2330	0.1890	0	Lab3/Delay2	Lab3/Register	clk	clk	create_clock -name clk -period 2 [get_ports clk]

b. Modify latency in various blocks in order to pass timing. This may be different than the suggestion in the tutorial. Explain your process.

I modified the latency of the multiplier in "subsystem1" from 1 to 2 cycles as suggested in the tutorial in order to meet timing for the Kintex device. It does not seem feasible to meet timing on the Artix device with the same 2ns clock period, even after significantly increasing latency of the multiplier and adder. Shown below are the results of the latency modification for the Kintex device.

c. Screen capture of timing analyzer results – passing – after modifications


Timing Analyzer: Lab3

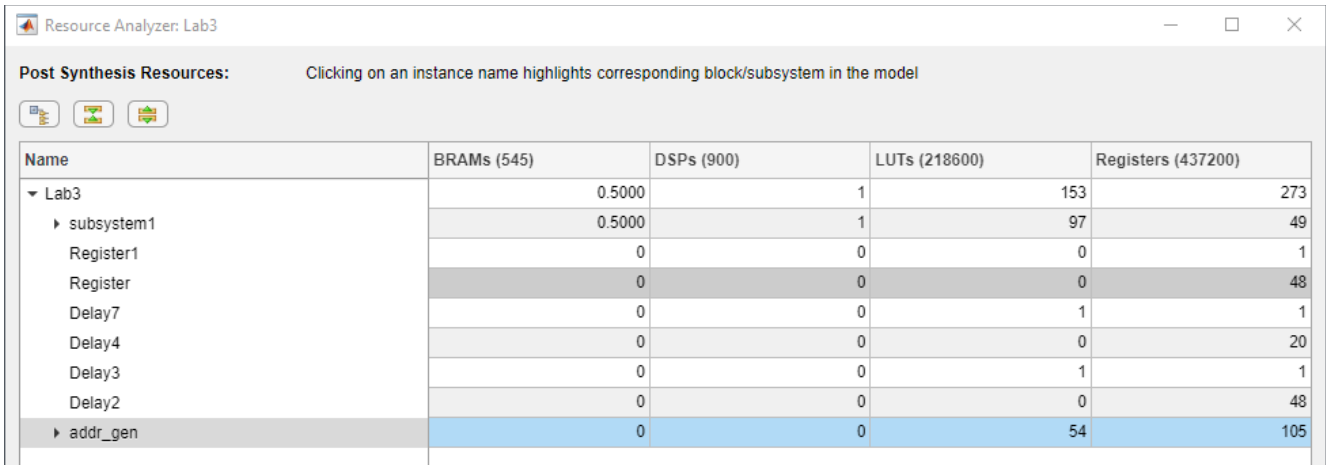
Post Synthesis Timing Paths: Clicking on an instance name highlights corresponding block/subsystem in the model

Violation type : setup — Select Columns — Status : PASSED

	Slack (ns)	Delay (ns)	Logic Delay (ns)	Routing Delay (ns)	Levels of Logic	Source	Destination	Source Clock	Destination Clock
1	0.0700	1.9230	1.3680	0.5550	14	Lab3/subsystem1/AddSub2	Lab3/subsystem1/AddSub2	clk	clk
2	0.5790	1.0410	0.6220	0.4190	0	Lab3/subsystem1/coef	Lab3/subsystem1/Mult	clk	clk
3	0.6100	1.3830	1.1090	0.2740	9	Lab3/addr_gen/Register4	Lab3/addr_gen/Register	clk	clk
4	0.7520	1.2380	0.8040	0.4340	3	Lab3/addr_gen/Register10	Lab3/addr_gen/Relational1	clk	clk
5	0.7620	0.9770	0.3560	0.6210	1	Lab3/Delay7	Lab3/Register	clk	clk
6	0.7860	0.6520	0.2330	0.4190	0	Lab3/Delay4	Lab3/subsystem1/coef	clk	clk
7	0.8560	1.1370	0.8390	0.2980	4	Lab3/addr_gen/Register	Lab3/addr_gen/AddSub1	clk	clk
8	1.1000	0.9030	0.9030	0	0	Lab3/Delay3	Lab3/Delay3	clk	clk
9	1.1890	0.8040	0.4930	0.3110	2	Lab3/addr_gen/Register	Lab3/addr_gen/Register	clk	clk
10	1.3470	0.4220	0.2330	0.1890	0	Lab3/addr_gen/Relational1	Lab3/Delay3	clk	clk
11	1.3580	0.6230	0.3560	0.2670	1	Lab3/Delay7	Lab3/Register1	clk	clk
12	1.3640	0.4810	0.2100	0.2710	0	Lab3/Delay3	Lab3/Delay7	clk	clk
13	1.3990	0.4460	0.2330	0.2130	0	Lab3/addr_gen/AddSub1	Lab3/addr_gen/Register10	clk	clk
14	1.3990	0.4460	0.2330	0.2130	0	Lab3/addr_gen/AddSub1	Lab3/addr_gen/Register3	clk	clk
15	1.4120	0.4330	0.2330	0.2000	0	Lab3/subsystem1/AddSub2	Lab3/Delay2	clk	clk
16	1.4230	0.4220	0.2330	0.1890	0	Lab3/addr_gen/Register3	Lab3/Delay4	clk	clk
17	1.4230	0.4220	0.2330	0.1890	0	Lab3/Delay4	Lab3/Delay4	clk	clk
18	1.4230	0.4220	0.2330	0.1890	0	Lab3/Delay2	Lab3/Register	clk	clk

Step 2

a. Screen capture of resource analyzer results for the Zynq Artix chip



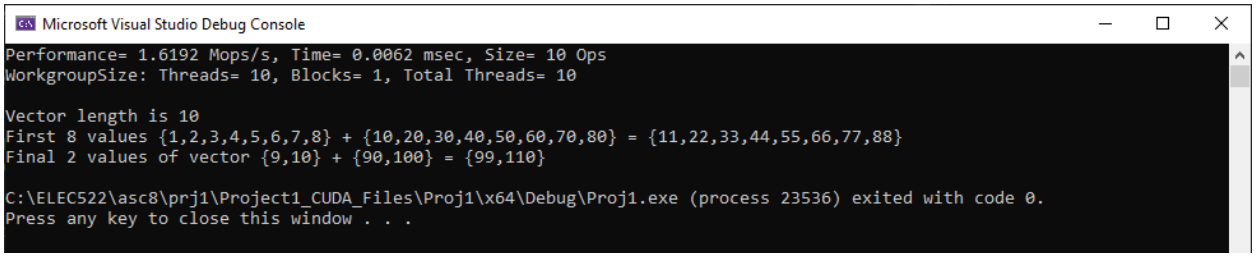
Name	BRAMs (545)	DSPs (900)	LUTs (218600)	Registers (437200)
Lab3	0.5000	1	153	273
subsystem1	0.5000	1	97	49
Register1	0	0	0	1
Register	0	0	0	48
Delay7	0	0	1	1
Delay4	0	0	0	20
Delay3	0	0	1	1
Delay2	0	0	0	48
addr_gen	0	0	54	105

b. Compare your results with the tutorial results and note any differences or if the same

For the Kintex part, increasing the multiplier latency from 1 cycle to 2 cycles indeed resolved the single timing issue, as expected from the tutorial.

CUDA Lab

a. Screen capture of debug results window for arraySize = 10, threads = size, blocks = 1 initial parameters



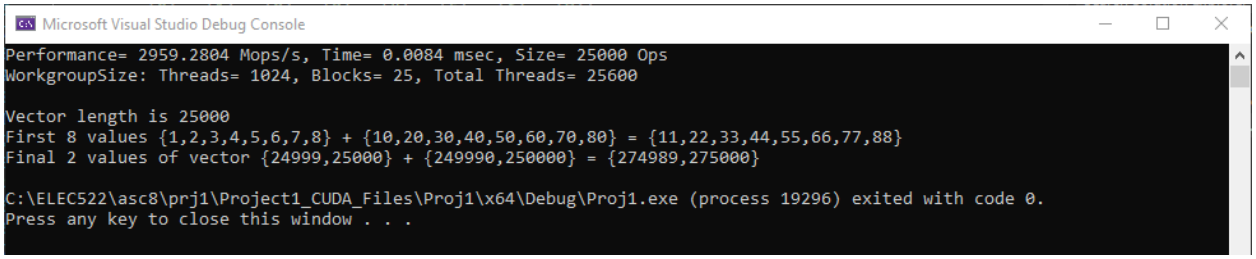
b. Include table of results for all of the arraySize values listed in the problem: 100, 200, 500, 750, 1000, 1250, 1500, 2000, 5000, 10000, 25000, and 64000. Table should record: Performance in Mops/s, Time, Size, Threads, Blocks, and Total Threads

Performance in Mops/s	Time	Size	Threads	Blocks	Total Threads
1.6192	0.0062	10	10	1	10
29.3427	0.0068	200	200	1	200
80.1282	0.0062	500	500	1	500
120.8119	0.0062	750	750	1	750
150.2404	0.0067	1000	1000	1	1000
159.4388	0.0078	1250	1024	2	2048
205.5921	0.0073	1500	1024	2	2048
245.098	0.0082	2000	1024	2	2048
645.6612	0.0077	5000	1024	5	5120
1509.6618	0.0066	10000	1024	10	10240
2959.2804	0.0084	25000	1024	25	25600
6024.0966	0.0106	64000	1024	63	64512

c. What happens at arraySize = 1250 with the original threads and blocks value? Explain.

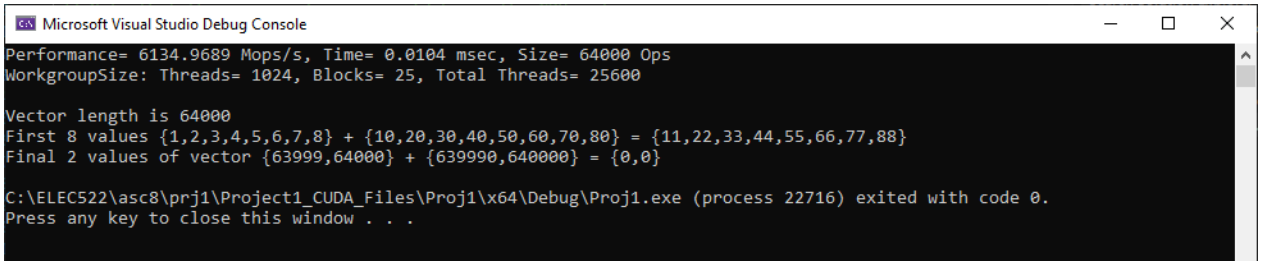
At arraySize = 1250, the maximum number of threads per block has been exceeded if the number of blocks is still set to 1024. The number of threads must be limited to 1024 at this point, and the number of blocks must be increased so that at least as many threads are available as the number of array elements involved.

d. Screen capture of the Debug results for arraySize = 25000



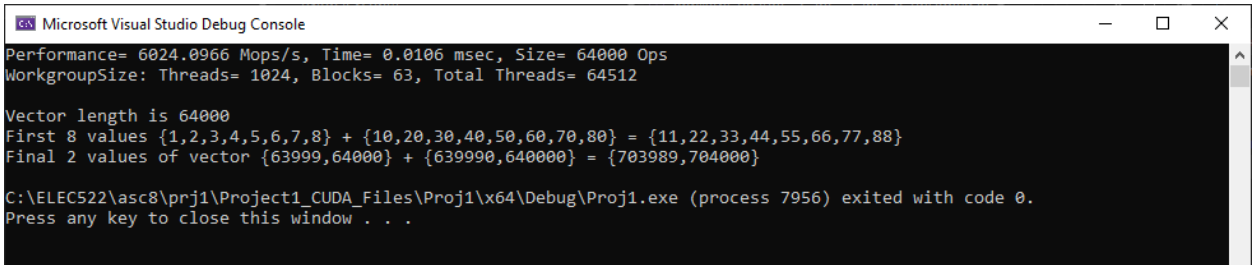
e. With the minimum threads and blocks sizes appropriate for arraySize = 25000, increase the arraySize to 64000. What are the values of the final two values of c? Explain.

When the arraySize is increased to 64000 but the number of blocks is not increased, the values of the sum at the end of the array are zero. This is because not enough threads are being used to cover all the indices in the array (only 25600 total, rather than 64000). The indexing on line 24 makes this clear. A screenshot displaying this condition is shown below.



f. Final two values of variable c at 64000 with minimum threads and blocks set for 25000. How did you fix these values?

To fix this problem, the number of blocks must be increased to 63. (I am assuming that the number of threads per block has been kept at the maximum of 1024.) The final correct values of c are shown below.



g. Analyze your table for the percentage increase in performance, time, and size

All changes are given as the increase relative to the values from the next lower size value.

Size	Size change	Performance change	Time change
10			
200	1900%	1712.17%	9.68%
500	150%	173.08%	-8.82%

750	50%	50.77%	0%
1000	33.33%	24.36%	8.06%
1250	25%	6.12%	16.42%
1500	20%	28.95%	-6.41%
2000	33.33%	19.22%	12.33%
5000	150%	163.43%	-6.1%
10000	100%	133.82%	-14.29%
25000	150%	96.02%	27.27%
64000	156%	103.57%	26.19%

From size 10 to size 64000, the performance in Mops/s increases 371941.54%, while the time required only increases 70.97%. The time does not even double despite the array size having increased by a factor of 6400. In general, the vector add for all array sizes tested takes no more than 11 microseconds (and less time for arrays smaller than 64000 elements), and can be variable. In the table of percentage-changes above, these small variations in time are observed between runs; these are to be expected due to variation in other system activity when such a short computation is being measured. A small decrease in the amount of time taken when the array size increases is not necessarily an indication of an anomaly. In summary, this computation scales well across the hardware available on the GPU, and the fact that we see huge improvements in performance with very little (if any) extra time taken indicates that computation on individual array elements is effectively being performed in parallel, without any significant issues with I/O bandwidth.

h. Include final Proj1_kernel.cu file

My final Proj1_kernel.cu file is included along with this submission on Canvas.