ESE5320 hw5

Wirte up by songqij

Answer

Q1

1.

```
root@u96v2-sbc-base-2020-2:~/ese5320/hw5/hls# g++ -O3 -mcpu=native -fno-tree-vectorize -o testbench Testbench.cpp MatrixMultiplication.cpp -std=c++11
root@u96v2-sbc-base-2020-2:~/ese5320/hw5/hls# ./testbench
Time taken for mmult:1.22072ms
TEST PASSED
```

The time for mmult is 1.22ms

2.

```
Starting C simulation ...
C:/Xilinx/Vitis_HLS/2020.2/bin/vitis_hls.bat S:/ese532_code/hw5/VISIT/Hw5/solution1/csim.tcl
INFO: [HLS 200-10] Running 'C:/Xilinx/Vitis_HLS/2020.2/bin/unwrapped/win64.o/vitis_hls.exe'
INFO: [HLS 200-10] For user 'songqij' on host 'mor200-05' (Windows NT_amd64 version 6.2) on Tue Oct 01 18:26:06 -0400 2024
INFO: [HLS 200-10] In directory 'S:/ese532_code/hw5/VISIT'
Sourcing Tcl script 'S:/ese532_code/hw5/VISIT/Hw5/solution1/csim.tcl'
INFO: [HLS 200-1510] Running: open_project Hw5
INFO: [HLS 200-10] Opening project 'S:/ese532_code/hw5/VISIT/Hw5'.
INFO: [HLS 200-1510] Running: set_top mmult
INFO: [HLS 200-1510] Running: add_files ../hls/MatrixMultiplication.cpp
INFO: [HLS 200-10] Adding design file '../hls/MatrixMultiplication.cpp' to the project
INFO: [HLS 200-1510] Running: add_files ../hls/MatrixMultiplication.h
INFO: [HLS 200-10] Adding design file '../hls/MatrixMultiplication.h' to the project
INFO: [HLS 200-1510] Running: add_files -tb ../hls/Testbench.cpp
INFO: [HLS 200-10] Adding test bench file '../hls/Testbench.cpp' to the project
INFO: [HLS 200-1510] Running: open_solution solution1 -flow_target vitis
INFO: [HLS 200-10] Opening solution 'S:/ese532_code/hw5/VISIT/Hw5/solution1'.
INFO: [SYN 201-201] Setting up clock 'default' with a period of 6.667ns.
INFO: [HLS 200-10] Setting target device to 'xczu3eg-sbva484-1-i'
INFO: [HLS 200-1505] Using flow_target 'vitis'
Resolution: For help on HLS 200-1505 see www.xilinx.com/cgi-bin/docs/rdoc?v=2020.2;t=hls+guidance;d=200-1505.html
INFO: [HLS 200-1464] Running solution command: config_compile -complex-mul-dsp=0
INFO: [HLS 200-1464] Running solution command: config_compile -pipeline_loops=0
INFO: [XFORM 203-1171] Pipeline the innermost loop with trip count more than 0 or its parent loop when its trip count is less than or equal 0.
INFO: [HLS 200-1464] Running solution command: config_interface -default_interface=kernel
INFO: [HLS 200-1464] Running solution command: config_interface -m_axi_alignment_byte_size=64
INFO: [HLS 200-1464] Running solution command: config_interface -m_axi_latency=64
INFO: [HLS 200-1464] Running solution command: config_interface -m_axi_max_widen_bitwidth=512
INFO: [HLS 200-1464] Running solution command: config_interface -m_axi_offset=slave
INFO: [HLS 200-1464] Running solution command: config_rtl -register_reset_num=3
INFO: [HLS 200-1510] Running: set_part xczu3eg-sbva484-1-i
INFO: [XFORM 203-1171] Pipeline the innermost loop with trip count more than 0 or its parent loop when its trip count is less than or equal 0.
INFO: [HLS 200-1510] Running: create_clock -period 150MHz -name default
INFO: [HLS 200-1510] Running: config_compile -pipeline_loops 0
INFO: [XFORM 203-1171] Pipeline the innermost loop with trip count more than 0 or its parent loop when its trip count is less than or equal 0.
INFO: [HLS 200-1510] Running: config_interface -m_axi_alignment_byte_size 64 -m_axi_latency 64 -m_axi_max_widen_bitwidth 512 -m_axi_offset slave
INFO: [HLS 200-1510] Running: config_rtl -register_reset_num 3
INFO: [HLS 200-1510] Running: set_directive_top -name mmult mmult
INFO: [HLS 200-1510] Running: csim_design -quiet
INFO: [SIM 211-2] *************** CSIM start ***************
INFO: [SIM 211-4] CSIM will launch GCC as the compiler.
   Compiling ../../../../../hls/MatrixMultiplication.cpp in debug mode
   Generating csim.exe
TEST PASSED
INFO: [SIM 211-1] CSim done with 0 errors.
INFO: [SIM 211-3] *************** CSIM finish ***************
INFO: [HLS 200-111] Finished Command csim_design CPU user time: 0 seconds. CPU system time: 0 seconds. Elapsed time: 5.003 seconds; current allocated memory: 159.328 MB.
Finished C simulation.
```

3.

Our application involves matrix multiplication. To verify the functionality of the code, we use test input, like how circuits are tested using signals. The Testbench.cpp handles this by providing two test matrices and comparing the actual output with the expected one. When they are the same, they pass the test, while not it fails.

4.

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
2904461	2912653	19.364 ms	19.419 ms	2904462	2912654	none

Detail

Instance

Loop

The expected latency is 19.364~19.419ms.

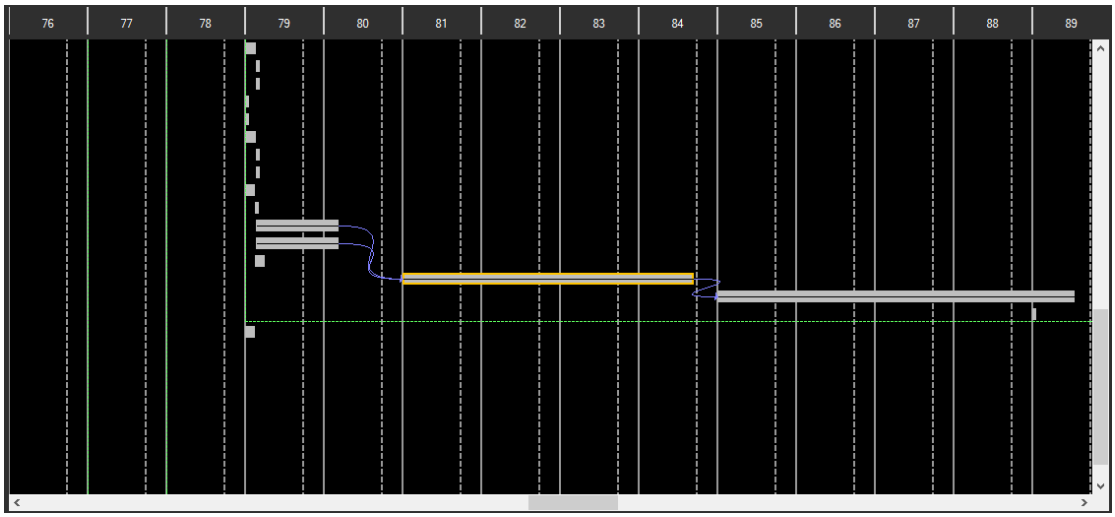
5.

Summary

Name	BRAM_18K	DSP	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	696	-
FIFO	-	-	-	-	-
Instance	60	3	3476	4457	-
Memory	16	-	0	0	-
Multiplexer	-	-	-	1064	-
Register	-	-	4168	-	-
Total	76	3	7644	6217	0
Available	432	360	141120	70560	0
Utilization (%)	17	~0	5	8	0

Name	BRAM	DSP	FF	LUT	URAM
Total	76	3	7644	6217	0

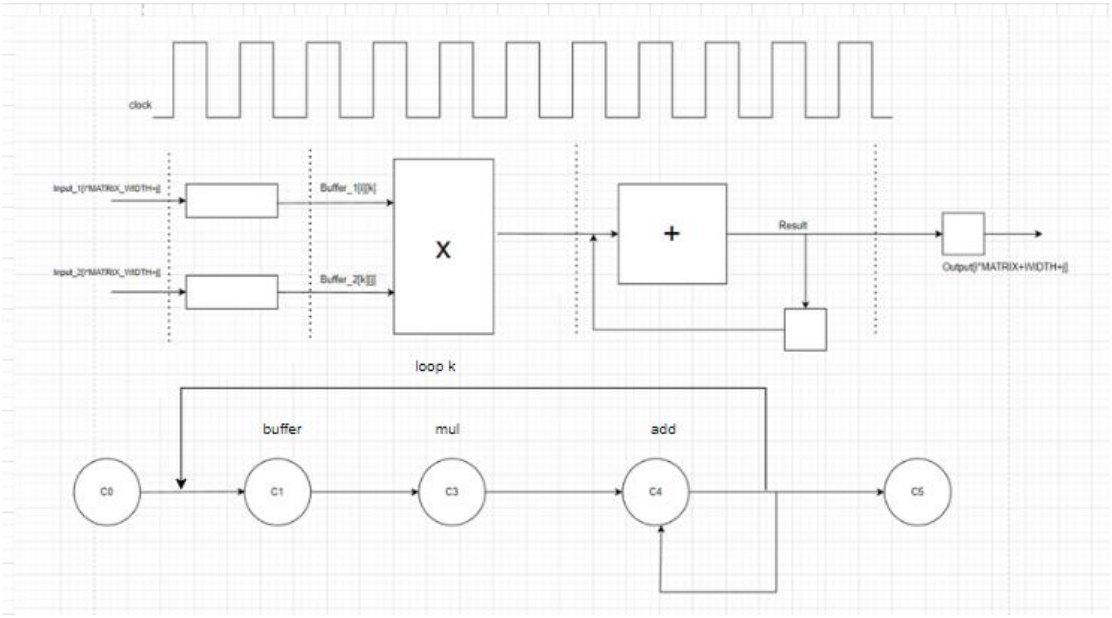
6.



The multiplication takes approximately 4 cycles.

7.

```
root@u96v2-sbc-base-2020-2:~/ese5320/hw5/hls# g++ -O3 -mcpu=native
root@u96v2-sbc-base-2020-2:~/ese5320/hw5/hls# ./testbench
Time taken for mmult:1.22072ms
TEST PASSED
```



8.

The are many factors that slow down the accelerator. Loops are unpipelined and reading and writing to the public memory also takes a relatively long time.

Q2

1.

```
Main_loop_k: for (int k = 0; k < MATRIX_WIDTH; k++) {
    #pragma HLS unroll factor=2
    Result += Buffer_1[i][k] * Buffer_2[k][j];
}
```

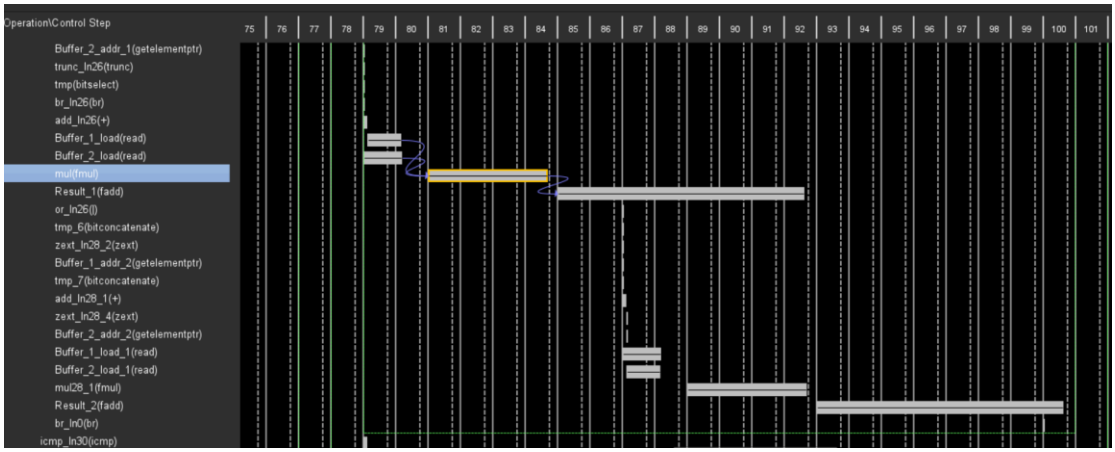
Performance & Resource Estimates

Modules Loops

Modules & Loops	Issue Type	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSP	FF	LUT	URAM
mmult		-	2912653	1.942E7	-	2912654	-	no	76	5	7731	5807	0
Init_loop_i		-	16512	1.100E5	258	-	64	no	-	-	-	-	-
Main_loop_i		-	2896000	1.931E7	45250	-	64	no	-	-	-	-	-

HW Interfaces

The latency hasn't changed from the original one.



We can see the difference from the previous one. We can see previous (non-unrolled) one take 10cycles to complete one loop, while unrolled one takes about 20 loops to complete 2 loops. Therefore, the latency is almost the same.

2.

```
Main_loop_k: for(int i = 0; k < MATRIX_WIDTH; k+=2){
    Result += Buffer_1[i][k] * Buffer_2[k][j];
    if(k+1 >= MATRIX_WIDTH) break;
    Result += Buffer_1[i][k+1]*Buffer_2[k+1][j];
    }
```

3.

We can see from the Schedule Viewer, every time the next Addition will executed after the previous one is finished. Therefore, the fadd has been shared.
Fadd.

4.

Target	Estimated	Uncertainty
6.67 ns	5.034 ns	1.80 ns

Performance & Resource Estimates

Modules Loops

Modules & Loops	Issue Type	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSP	FF	LUT	URAM
mmult		-0.16	2144653	1.430E7	-	2144654	-	no	76	8	13431	10061	0
Init_loop_i		-	16512	1.100E5	258	-	64	no	-	-	-	-	-
Main_loop_i		-	2128000	1.419E7	33250	-	64	no	-	-	-	-	-

$$6.67 - (5.034 + 1.8) = -0.164$$

It gets a negative slack time, which means that actually we need more times and the

clock frequency is too high. A negative slack of -0.16 nanoseconds means that the signal transmission time exceeds the time allocated for the timing path in the design by 0.16 nanoseconds, indicating that the clock cycle is insufficient to accommodate the signal propagation time. It cause the **timing violation.**

5.

⊟ Timing

⊟ Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	7.300 ns	2.70 ns

⊟ Latency

⊟ Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
1345933	1354125	13.459 ms	13.541 ms	1345934	1354126	none

⊟ Detail

The expected latency is 13.459~13.541ms

6.

Utilization Estimates

⊟ Summary

Name	BRAM_18K	DSP	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	1808	-
FIFO	-	-	-	-	-
Instance	60	8	3567	4080	-
Memory	16	-	0	0	-
Multiplexer	-	-	-	4046	-
Register	-	-	9228	-	-
Total	76	8	12795	9934	0
Available	432	360	141120	70560	0
Utilization (%)	17	2	9	14	0

Name	BRAM	DSP	FF	LUT	URAM
Total	76	8	12795	9934	0

7.

Latency: Each floating-point addition must complete before the next one begins, which increases the overall latency.

Non-parallel execution: Floating-point additions cannot be executed in parallel, which limits the potential throughput of the system.

8.

For 100MHz unroll:

It take 17% of the BRAM_18K

For 150MHz non-unroll:

It take 17% of the BRAM_18K

Therefore these two ways can takes the same copies. However, the 100MHz unroll latency is lower.

I will choose unroll 100MHz.

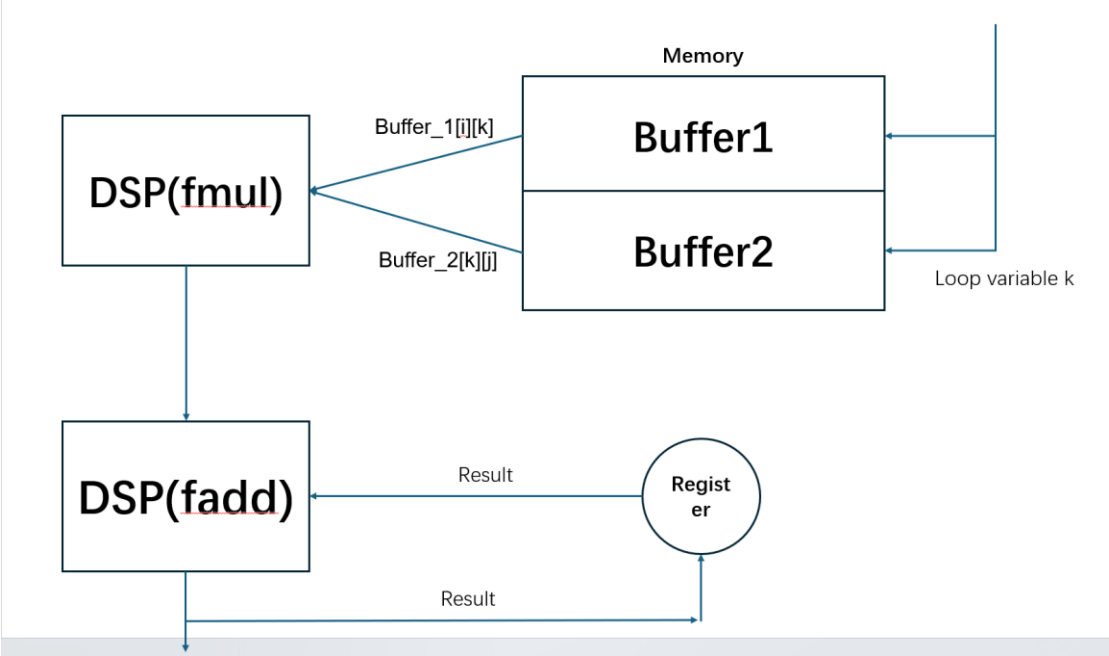
Q3

1.

Modules & Loops	Issue Type	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSP	FF	LUT	URAM
mmult	II Violation	-	33554	2.240E5	-	33555	-	no	300	80	27254	17698	0
Init_loop_i		-	16512	1.100E5	258	-	64	no	-	-	-	-	-
Main_loop_i_Main_loop_j	II Violation	-	16901	1.130E5	522	4	4096	yes	-	-	-	-	-

The interval is 4

2.



3.

$$\frac{64 * 4bytes}{4} = 64$$

It should be 64 per second.

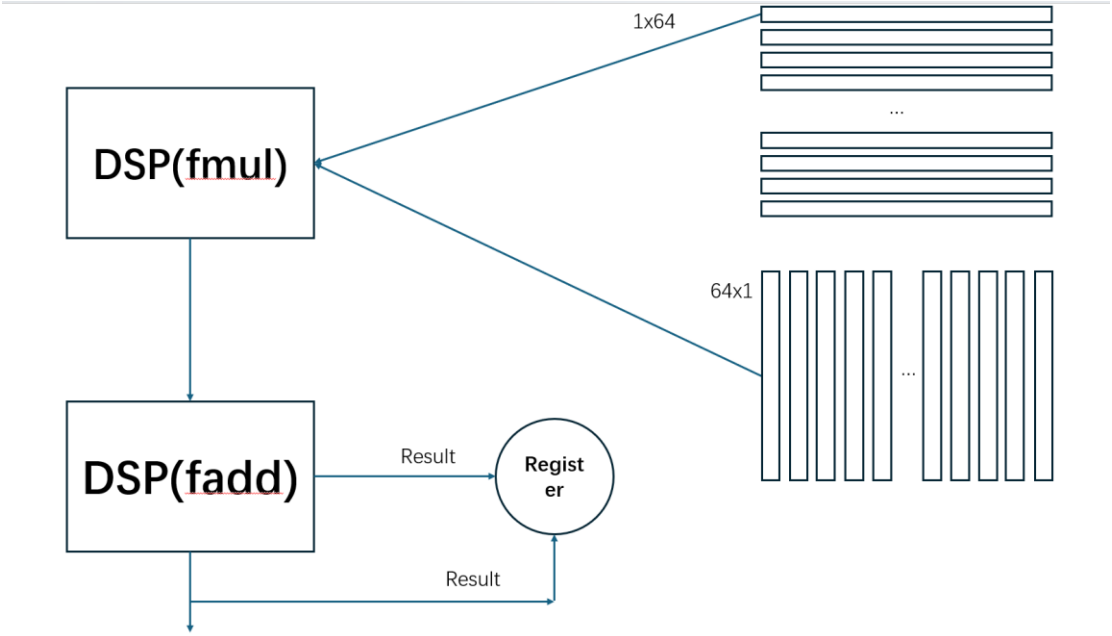
4.

First of all, the loading speed is too low, every time it could only load 16bytes and not

enough for the execution.
And for adding, it adds sequentially, so that we could not achieve the II of 1.

5.

I divided the buffer_1 which is 64*64 into 64* 1x64
I divided the buffer_2 which is 64*64 into 64* 64x1
And in one loop iteration will multiply 1x64 and 64x1 these two buffers.



6.

```cpp
#include "MatrixMultiplication.h"

void mmult(const matrix_type Input_1[MATRIX_WIDTH * MATRIX_WIDTH],
        const matrix_type Input_2[MATRIX_WIDTH * MATRIX_WIDTH],
        matrix_type Output[MATRIX_WIDTH * MATRIX_WIDTH]) {
#pragma HLS INTERFACE m_axi port=Input_1 bundle=aximm1
#pragma HLS INTERFACE m_axi port=Input_2 bundle=aximm2
#pragma HLS INTERFACE m_axi port=Output bundle=aximm1
    matrix_type Buffer_1[MATRIX_WIDTH][MATRIX_WIDTH];
    matrix_type Buffer_2[MATRIX_WIDTH][MATRIX_WIDTH];
    #pragma HLS array_partition variable=Buffer_1 complete dim=2
    #pragma HLS array_partition variable=Buffer_2 complete dim=1

    Init_loop_i: for (int i = 0; i < MATRIX_WIDTH; i++)
        Init_loop_j: for (int j = 0; j < MATRIX_WIDTH; j++) {
            #pragma HLS pipeline II=1
            Buffer_1[i][j] = Input_1[i * MATRIX_WIDTH + j];
            Buffer_2[i][j] = Input_2[i * MATRIX_WIDTH + j];
        }

    Main_loop_i: for (int i = 0; i < MATRIX_WIDTH; i++)

        Main_loop_j: for (int j = 0; j < MATRIX_WIDTH; j++) {
        #pragma HLS pipeline II=1

            matrix_type Result = 0;
            Main_loop_k: for (int k = 0; k < MATRIX_WIDTH; k++) {
                //#pragma HLS unroll
                Result += Buffer_1[i][k] * Buffer_2[k][j];
            }
            Output[i * MATRIX_WIDTH + j] = Result;
        }
}
```

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	6.67 ns	4.867 ns	1.80 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
8852	8852	59.016 us	59.016 us	8853	8853	none

Detail

Instance

Loop

The expected latency is 0.059016ms

7.

Utilization Estimates

Summary

Name	BRAM_18K	DSP	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	2122	-
FIFO	-	-	-	-	-
Instance	60	320	31172	28170	-
Memory	128	-	0	0	-
Multiplexer	-	-	-	4060	-
Register	-	-	12970	160	-
Total	188	320	44142	34512	0
Available	432	360	141120	70560	0
Utilization (%)	43	88	31	48	0

Name	BRAM	DSP	FF	LUT	URAM
Total	188	320	44142	34512	0

Yes, it is pretty consuming on DSP usage as it gets the 88% of Utilization.
And for BRAM, flip flop and LUT all reached nearly 50% of Utilization.

8.

```
#include "MatrixMultiplication.h"

void mmult(const matrix_type Input_1[MATRIX_WIDTH * MATRIX_WIDTH],
        const matrix_type Input_2[MATRIX_WIDTH * MATRIX_WIDTH],
        matrix_type Output[MATRIX_WIDTH * MATRIX_WIDTH]) {
#pragma HLS INTERFACE m_axi port=Input_1 bundle=aximm1
#pragma HLS INTERFACE m_axi port=Input_2 bundle=aximm2
#pragma HLS INTERFACE m_axi port=Output bundle=aximm1
    matrix_type Buffer_1[MATRIX_WIDTH][MATRIX_WIDTH];
    matrix_type Buffer_2[MATRIX_WIDTH][MATRIX_WIDTH];
    #pragma HLS array_partition variable=Buffer_1 complete dim=2
    #pragma HLS array_partition variable=Buffer_2 complete dim=1

    Init_loop_i: for (int i = 0; i < MATRIX_WIDTH; i++)
        Init_loop_j: for (int j = 0; j < MATRIX_WIDTH; j++) {
            #pragma HLS pipeline II=1
            Buffer_1[i][j] = Input_1[i * MATRIX_WIDTH + j];
            Buffer_2[i][j] = Input_2[i * MATRIX_WIDTH + j];
        }

    Main_loop_i: for (int i = 0; i < MATRIX_WIDTH; i++)

        Main_loop_j: for (int j = 0; j < MATRIX_WIDTH; j++) {
        #pragma HLS pipeline II=1

            matrix_type Result = 0;
            Main_loop_k: for (int k = 0; k < MATRIX_WIDTH; k++) {
                //#pragma HLS unroll
                Result += Buffer_1[i][k] * Buffer_2[k][j];
            }
            Output[i * MATRIX_WIDTH + j] = Result;
        }
}
```

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	6.67 ns	4.867 ns	1.80 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
8852	8852	59.016 us	59.016 us	8853	8853	none

Detail

Instance

Loop

Utilization Estimates

Summary

Name	BRAM_18K	DSP	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	2122	-
FIFO	-	-	-	-	-
Instance	60	320	31172	28170	-
Memory	128	-	0	0	-
Multiplexer	-	-	-	4060	-
Register	-	-	12970	160	-
Total	188	320	44142	34512	0
Available	432	360	141120	70560	0
Utilization (%)	43	88	31	48	0

General Information

Date: Wed Oct 2 20:06:52 EDT 2024
Version: 2020.2 (Build 3064766 on Wed Nov 18 09:12:47 MST 2020)
Project: HW5
Status: Pass

Solution: solution1 (Vitis Kernel Flow Target)
Product family: zynquplus
Target device: xczu3eg-sbva484-1-i

Cosim Options

Tool: Vivado XSIM

RTL: Verilog

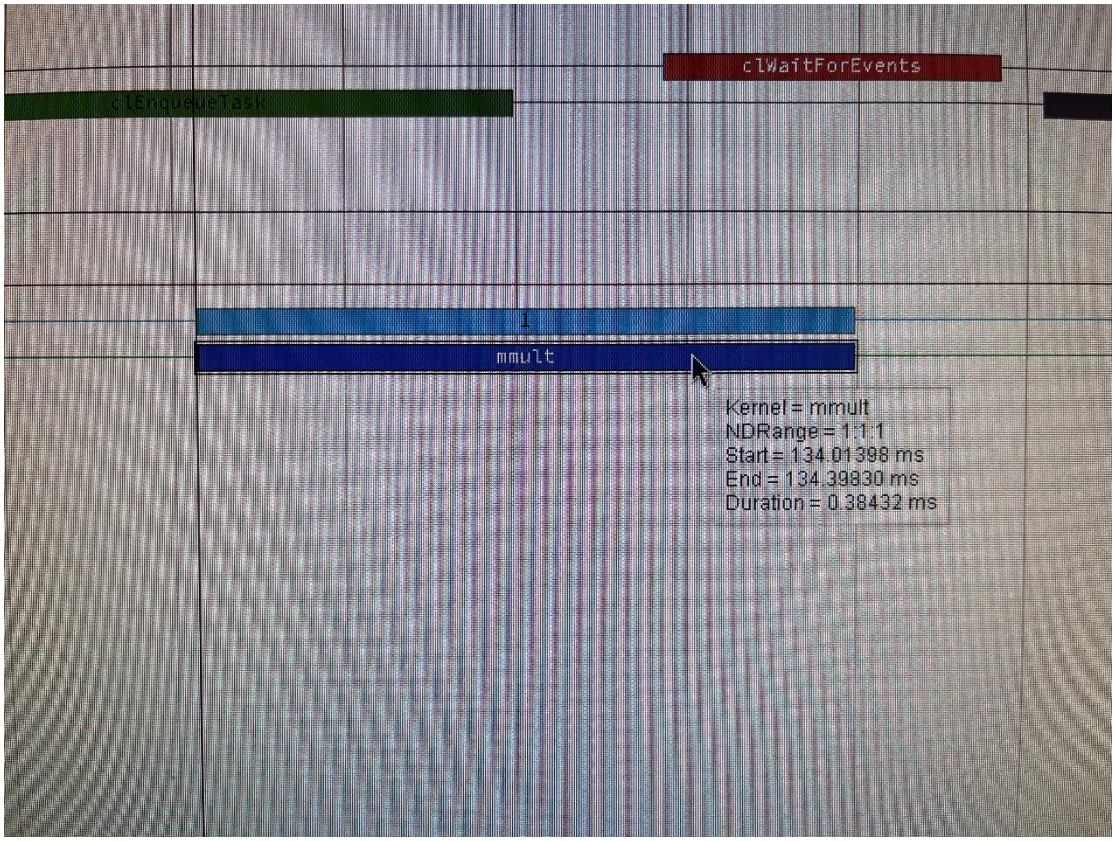
Performance Estimates

9.Done

Q4

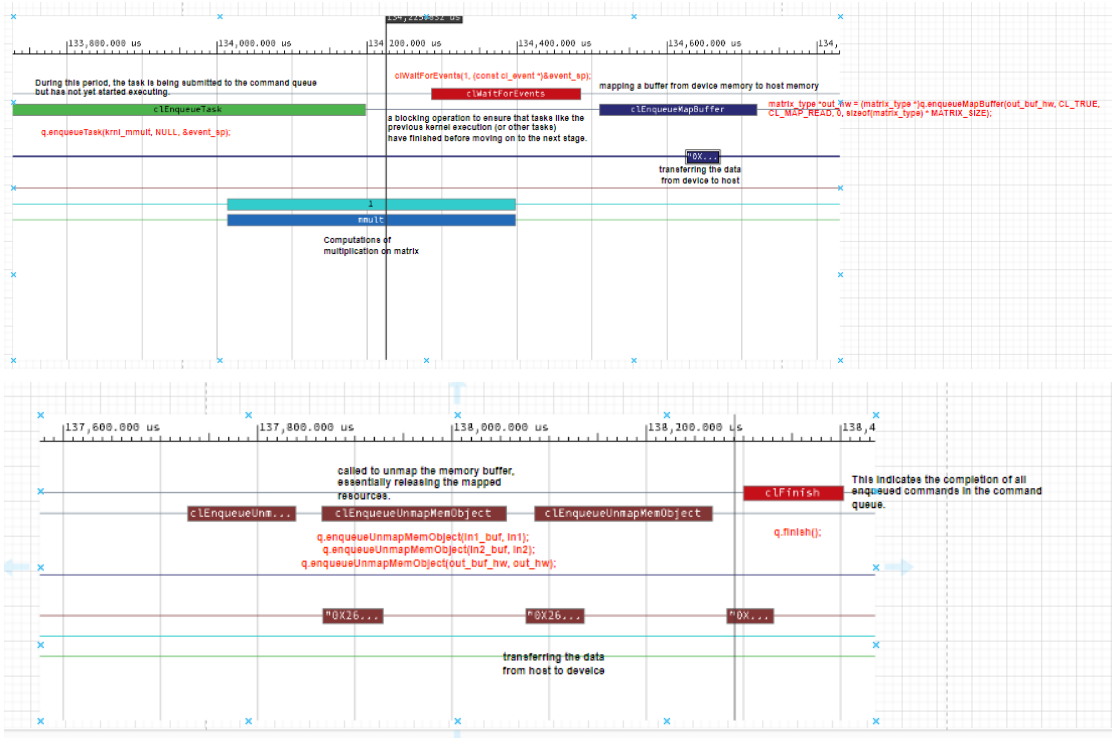
1.Done

2.



The latency of mmult is 0.38432ms

3.



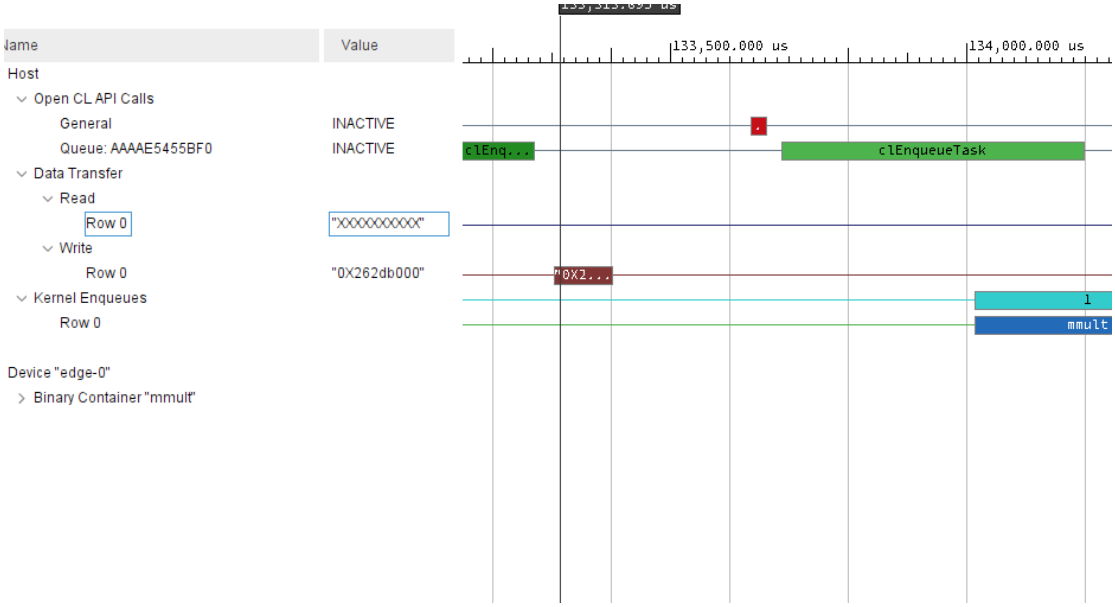
Q5.

1.

$$S_{fpga} = \frac{T_{seq}}{T_{fpga}} = \frac{1.22072ms}{0.38432ms} = 3.17$$

2.

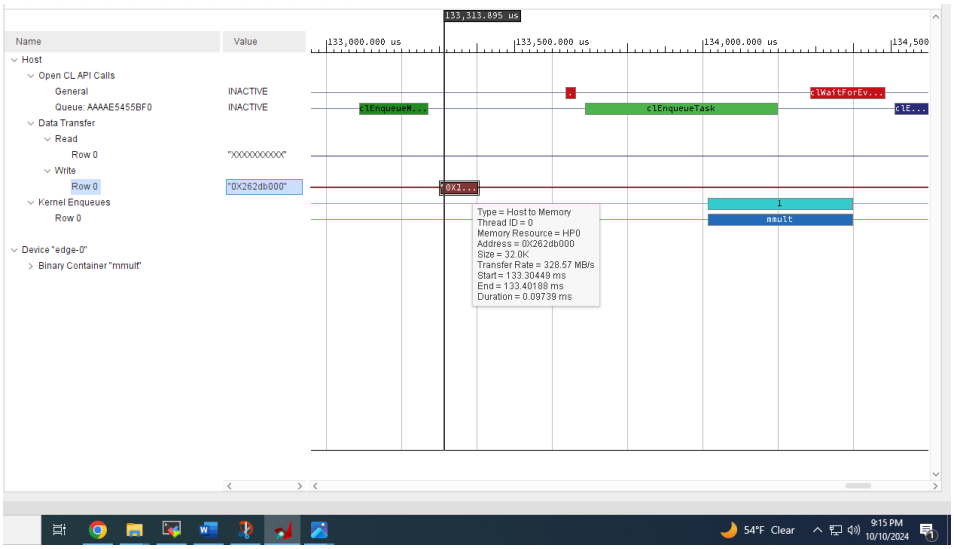
For we are calculating T setup, we just add up all the time before mmult including space time.



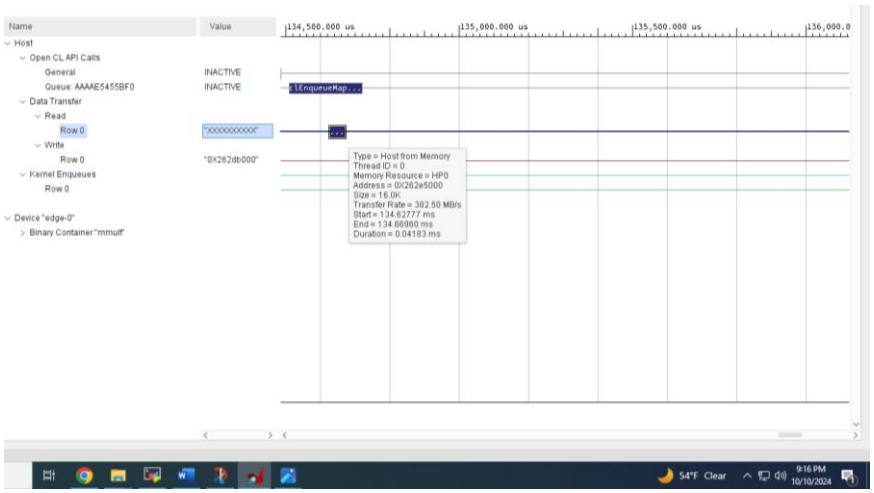
$$T_{setup} = 133.5ms$$

3.

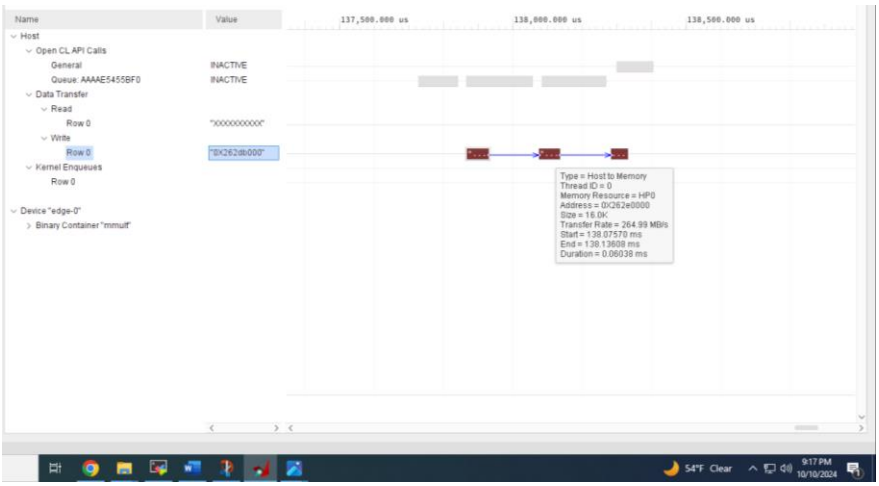
For calculating the $T_{transfer}$, just add up all the host from memory and host to memory.



$$T_{host\ to\ memory} = 0.09739ms$$



$$T_{host\ from\ memory} = 0.04183ms$$



$$T_{host\ to\ memory} = 0.06038ms + 0.06190ms + 0.04780ms = 0.178ms$$

Therefore, $T_{transfer} = 0.178ms + 0.09739ms + 0.04183ms = 0.317ms$

4.

$$T_{accel} = T_{setup} + T_{transfer} + T_{fpga} = 133.5ms + 0.317ms + 0.38ms = 134.197ms$$

$$\frac{T_{seq}}{T_{accel}} > 1$$

T_{accel} should be larger than 134.197ms.

5.

For T_{seq}:

$$T_{seq} = N^3 + N^2$$

6.

For the main loop j pipelined and loop k unrolled the T_{fpga} should be:

$$T_{fpga} = N^2 + N^2 = 2N^2$$

7.

The transferring time should be the reading adds to the writing part.

The reading time with two buffers are $T_{read} = 2N^2$

The writing time with the buffer is $T_{write} = N^2$

$$T_{transfer} = 2N^2 + N^2 = 3N^2$$

8.

$$T_{accel} = T_{setup} + T_{transfer} + T_{fpga} = 134ms + 5N^2$$

$$T_{seq} = T_{accel}$$

$$N^3 + N^2 = 134ms + 5N^2$$

$$N = 7$$

$$7 * 64 = 448$$

9.

$$\frac{T_{seq}}{10} = T_{setup} + T_{transfer} + T_{fpga} = 134ms + 5N^2$$

$$N = 50$$

$$50 * 64 = 3200$$

10.

$$S_{fpga} = \frac{T_{seq}}{T_{fpga}} = \frac{N^3 + N^2}{2N^2} = 1600$$

11.

$$\frac{T_{seq}}{10} = T_{setup} + k * (T_{transfer} + T_{fpga})$$

$$\frac{N^3 + N^2}{10} = 134ms + 10^6 * 5N^2$$

$$N = 5 * 10^7$$

$$5 * 10^7 * 64 = 3.2 * 10^9$$

Q6.

For question one we measure the baseline time for the matrix multiplication.

For question two we do the unroll of the main loop k and do the optimization.

For question three we do the pipeline of the main loop j and the initial loop j, and get the final optimization.

2.

	latency	DSP	BlockRAMs
1	19.419ms	3	76
2	13.541ms	8	76
3	0.059016ms	320	188