CS5222 Advanced Architectures

Project 2 Part 1 Report

FPGA Lab

Nguyen Dang Phuc Nhat (A0184583U)

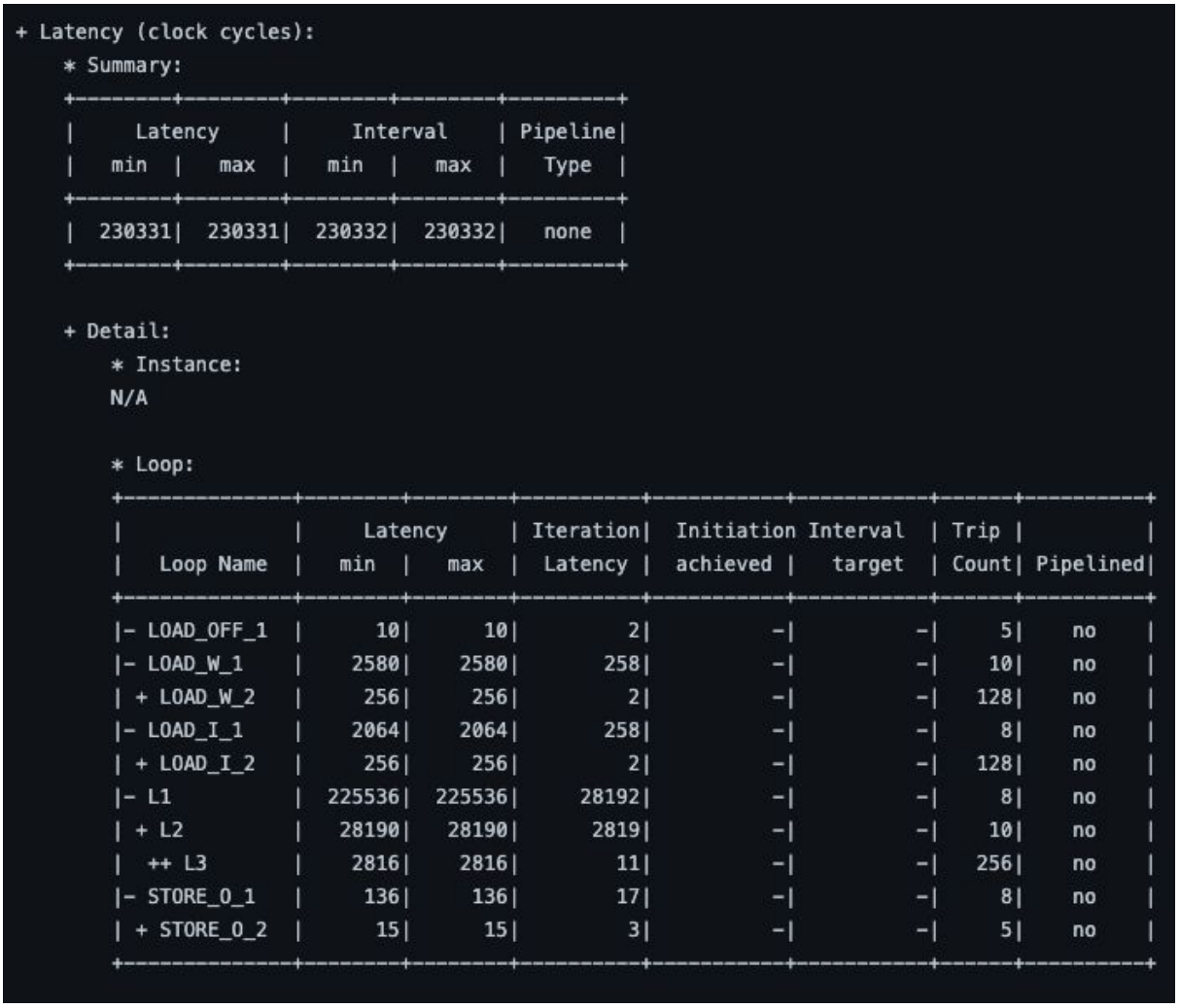
Table of Contents

1 Task A

Running the baseline code, the results are as follows:

1. Latency

The latency is 230331 cycles for the baseline model. This is different from the expected number 209851. However, the difference here is irrelevant as we are interested in speedup in subsequent tasks.



```
+ Latency (clock cycles):
    * Summary:
    +--------+--------+--------+--------+----------+
    |     Latency     |     Interval    | Pipeline|
    |   min  |   max  |   min  |   max  |   Type  |
    +--------+--------+--------+--------+----------+
    |  230331|  230331|  230332|  230332|   none  |
    +--------+--------+--------+--------+----------+

    + Detail:
        * Instance:
        N/A

        * Loop:
        +--------------+--------+--------+----------+-----------+-----------+------+----------+
        |              |     Latency     | Iteration|  Initiation Interval  | Trip |          |
        |   Loop Name  |   min  |   max  |  Latency |  achieved |   target  | Count| Pipelined|
        +--------------+--------+--------+----------+-----------+-----------+------+----------+
        |- LOAD_OFF_1  |      10|      10|         2|          -|          -|     5|    no    |
        |- LOAD_W_1    |    2580|    2580|       258|          -|          -|    10|    no    |
        | + LOAD_W_2   |     256|     256|         2|          -|          -|   128|    no    |
        |- LOAD_I_1    |    2064|    2064|       258|          -|          -|     8|    no    |
        | + LOAD_I_2   |     256|     256|         2|          -|          -|   128|    no    |
        |- L1          |  225536|  225536|     28192|          -|          -|     8|    no    |
        | + L2         |   28190|   28190|      2819|          -|          -|    10|    no    |
        |  ++ L3       |    2816|    2816|        11|          -|          -|   256|    no    |
        |- STORE_O_1   |     136|     136|        17|          -|          -|     8|    no    |
        | + STORE_O_2  |      15|      15|         3|          -|          -|     5|    no    |
        +--------------+--------+--------+----------+-----------+-----------+------+----------+
```

2. Utilization

Despite a few minor differences, all components' utilization is under 5% just like the write-up's information.

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	537
FIFO	-	-	-	-
Instance	0	5	384	751
Memory	16	-	0	0
Multiplexer	-	-	-	558
Register	-	-	779	-
Total	16	5	1163	1846
Available	280	220	106400	53200
Utilization (%)	5	2	1	3

2 Task B

I started by pipelining the L3 loop only (as per the guide's suggestion). This obviously could not provide enough speedup. The resulting latency is 107995 cycles, which is a 2.13x speedup.

In the optimised design, L2 is pipelined. This caused L3 to be unrolled completely. The resulting pipelined loop is therefore called L1-L2. The initiation interval (II) achieved for this pipeline is 128, of course very far from the target II=1.

To increase performance further, I also pipelined LOAD_OFF_1, LOAD_W_2, LOAD_I_2, and STORE_O_2. In short, all innermost loops are pipelined.

The latency of this design is 13841 cycles, which is a 16.64x speedup.

+ Latency (clock cycles):
    * Summary:

Latency		Interval		Pipeline
min	max	min	max	Type
13840	13840	13841	13841	none

    + Detail:
        * Instance:
        N/A

        * Loop:

Loop Name	Latency		Iteration	Initiation Interval		Trip	Pipelined
	min	max	Latency	achieved	target	Count	
|- LOAD_OFF_1	5	5	2	1	1	5	yes
|- LOAD_W_1	1310	1310	131	-	-	10	no
| + LOAD_W_2	128	128	2	1	1	128	yes
|- LOAD_I_1	1048	1048	131	-	-	8	no
| + LOAD_I_2	128	128	2	1	1	128	yes
|- L1_L2	11398	11398	1287	128	1	80	yes
|- STORE_O_1	72	72	9	-	-	8	no
| + STORE_O_2	6	6	3	1	1	5	yes

As expected, this design utilized significantly more hardware. 72% of LUT is utilized - much more than in the baseline design.

Name	BRAM_18K	DSP48E	FF	LUT
|DSP	-	-	-	-
|Expression	-	-	0	26037
|FIFO	-	-	-	-
|Instance	0	10	732	1462
|Memory	16	-	0	0
|Multiplexer	-	-	-	4793
|Register	-	-	24611	6496
|Total	16	10	25343	38788
|Available	280	220	106400	53200
|Utilization (%)	5	4	23	72

3 Task C

The design is built upon the one in Task B.

Partitioning in_buf only provided no speedup, but instead reduced the hardware utilization. Therefore, I partitioned weight_buf as well, since this array is also used heavily in the computation.

Partitioning each into 4 arrays gives a latency of 6257 cycles - 36.81x speedup.

The most partitions I can use is 8 for each buffer. With this design, the latency is 4992 cycles, which means a 46.14x speedup.

This design reduces the II of the L1-L2 pipeline significantly, from 128 to 16.

```
+ Latency (clock cycles):
    * Summary:
    +------+------+------+------+---------+
    |   Latency   |   Interval  | Pipeline|
    |  min |  max |  min |  max |   Type  |
    +------+------+------+------+---------+
    |  4992|  4992|  4993|  4993|   none  |
    +------+------+------+------+---------+

    + Detail:
        * Instance:
        N/A

        * Loop:
        +--------------+------+------+----------+----------+----------+------+----------+
        |              |   Latency   | Iteration|  Initiation Interval  | Trip |          |
        |   Loop Name  |  min |  max |  Latency |  achieved |   target  | Count| Pipelined|
        +--------------+------+------+----------+----------+----------+------+----------+
        |- LOAD_OFF_1  |     5|     5|         2|         1|         1|     5|    yes   |
        |- LOAD_W_1    |  1310|  1310|       131|         -|         -|    10|    no    |
        | + LOAD_W_2   |   128|   128|         2|         1|         1|   128|    yes   |
        |- LOAD_I_1    |  1048|  1048|       131|         -|         -|     8|    no    |
        | + LOAD_I_2   |   128|   128|         2|         1|         1|   128|    yes   |
        |- L1_L2       |  2550|  2550|      1287|        16|         1|    80|    yes   |
        |- STORE_O_1   |    72|    72|         9|         -|         -|     8|    no    |
        | + STORE_O_2  |     6|     6|         3|         1|         1|     5|    yes   |
        +--------------+------+------+----------+----------+----------+------+----------+
```

Hardware utilization is higher than Task B, as expected. It is a bit surprising that LUT utilization is now lower than Task B.

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	3309
FIFO	-	-	-	-
Instance	0	80	5604	11416
Memory	36	-	0	0
Multiplexer	-	-	-	6324
Register	-	-	32355	14129
Total	36	80	37959	35178
Available	280	220	106400	53200
Utilization (%)	12	36	35	66

To further illustrate the higher hardware requirement, this design requires 16 floating-point adders and 16 floating-point multipliers.

4 Task D

The designs are based on Task C final design.

By performing binary search, I could find that the maximum batch size is 256.

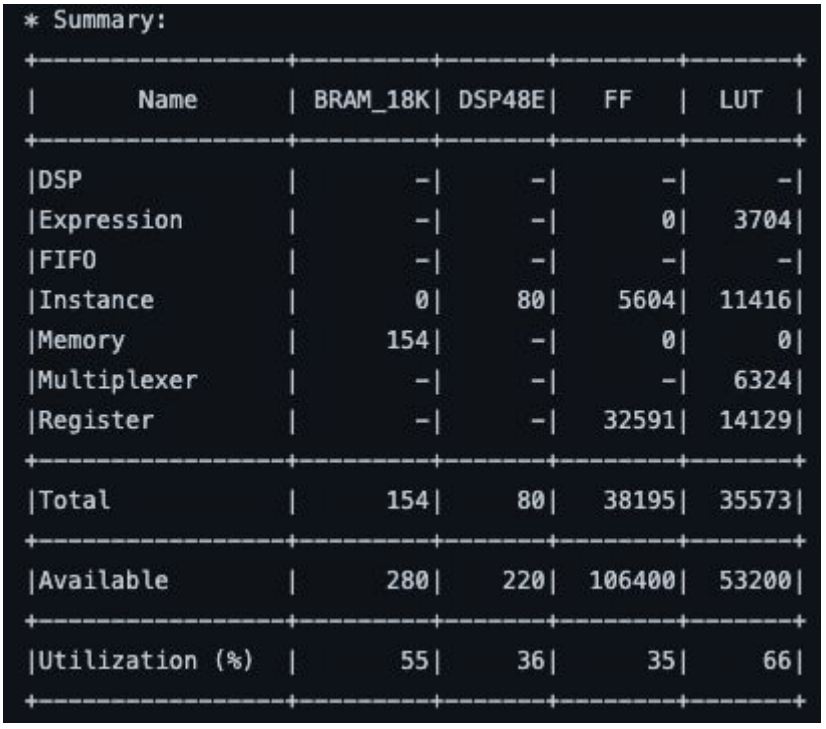
This design gives the overall latency of 79392 cycles. Without normalizing the batch size, this is a 2.9x speedup.

+ Latency (clock cycles):
    * Summary:

Latency		Interval		Pipeline
min	max	min	max	Type
79392	79392	79393	79393	none

After normalization with respect to batch size, the speedup is 92.84x.

Compared to Task C, only BRAM has higher utilization - 55% vs 12%.



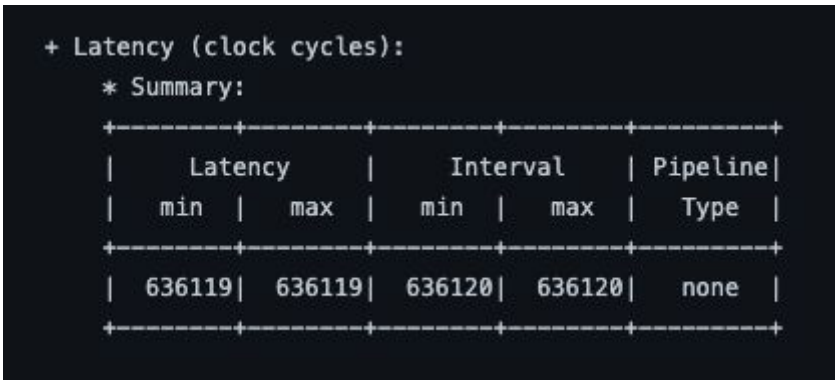
* Summary:

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	3704
FIFO	-	-	-	-
Instance	0	80	5604	11416
Memory	154	-	0	0
Multiplexer	-	-	-	6324
Register	-	-	32591	14129
Total	154	80	38195	35573
Available	280	220	106400	53200
Utilization (%)	55	36	35	66

5 Task E

By implementing tiling, the overall latency goes up to 636119 cycles.

With normalization, this gives a 92.7x speedup. The number is quite close to that in Task D. It shows that batch size really does amortize the latency.



+ Latency (clock cycles):

* Summary:

Latency		Interval		Pipeline
min	max	min	max	Type
636119	636119	636120	636120	none

With tiling implemented, the hardware utilization is reduced. BRAM utilization is lower than Task D, but lower than Task C.

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	3710
FIFO	-	-	-	-
Instance	0	80	5604	11416
Memory	86	-	0	0
Multiplexer	-	-	-	6346
Register	-	-	32639	14129
Total	86	80	38243	35601
Available	280	220	106400	53200
Utilization (%)	30	36	35	66

6 Task F

My intended design partitions each of in_buf and weight_buf into 4 arrays. However, due to compiling error, I had to scale it down further, to 2 arrays each.

Therefore, the speedup is not as high as the reported number in the write-up.

Still, the FPGA in my case could achieve a 4.2x speedup compared to CPU.

The classification accuracy is 86.96%.

```
FPGA accuracy: 13.04% validation error
CPU accuracy:  13.04% validation error
FPGA has a 4.21x speedup
```