CS5222 Project 2
Custom Acceleration with FPGAs

Shen Jiamin
A0209166A
shen_jiamin@u.nus.edu

March 6, 2022

Abstract

In this project, I'm going to port the lab to **PYNQ 2.7** and **Vivado/Vitis 2020.2**. The experiment is done on ASUS RS500-E8-PS4 V2, with operating system Ubuntu 20.04.4 LTS (GNU/Linux 5.4.0-100-generic x86_64).

1 Matrix Multiplication Pipeline Optimization in HLS

1.1 Understanding the baseline matrix multiply (background)

For Vitis 2020.2, the command used should be

```
$ vitis_hls -f hls.tcl
```

The report generated by HLS (as in Table 1) shows that some pipelining has already been done automatically by Vitis HLS. After inspecting the migration guide, I added two lines in the `hls.tcl`:

```
config_compile -pipeline_loops 0
set_clock_uncertainty 12.5%
```

The performance and utilization estimates of the two profiles, together with those for the following profiles, are reported in Table 3. The new loop details is as Table 2. It turns out that the overall performance is a little bit worse than documented. This is because every iteration in L3 loop takes 11 cycles and thus 2816 cycles in total to perform a single inner product.

2 Part 2: Fixed-Point Optimizations (30 marks)

1. the fixed-point validation accuracy reported by mnist.py after you've tweaked the SCALE factor.
2. the design latency in cycles

Table 1: Loop details for baseline with automatic pipelining

Loop Name	Latency (cycles)		Iteration Latency	Initiation Interval		Trip Count	Pipelined
	min	max		achieved	target		
- LOAD_OFF_1	5	5	1	1	1	5	yes
- LOAD_W_1_LOAD_W_2	1280	1280	1	1	1	1280	yes
- LOAD_I_1_LOAD_I_2	1024	1024	1	1	1	1024	yes
- L1_L2	82800	82800	1035	-	-	80	no
+ L3	1031	1031	12	4	1	256	yes
- STORE_O_1_STORE_O_2	42	42	4	1	1	40	yes

Table 2: Loop details for baseline without automatic pipelining

Loop Name	Latency (cycles)		Iteration Latency	Initiation Interval		Trip Count	Pipelined
	min	max		achieved	target		
- LOAD_OFF_1	5	5	1	-	-	5	no
- LOAD_W_1	1300	1300	130	-	-	10	no
+ LOAD_W_2	128	128	1	-	-	128	no
- LOAD_I_1	1040	1040	130	-	-	8	no
+ LOAD_I_2	128	128	1	-	-	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

Table 3: Performance and utilization estimates for `mmult_float`

	Profile	Latency (cycles)		Latency (ms)		Interval (cycles)		Pipeline Type
		min	max	min	max	min	max	
1.1	Baseline (AutoPipe)	85160	85160	1.236	1.236	85161	85161	none
1.1	Baseline (NoPipe)	228022	228022	2.280	2.280	228023	228023	none
??	L3 Pipelining	85286	85286	1.238	1.238	85287	85287	none
??	L2 Pipelining	13759	13759	0.138	0.138	13760	13760	none
??	L1 Pipelining (1WnR)	6193	6193	0.062	0.062	6194	6194	none
??	L1 Pipelining (T2P)	5953	5953	0.060	0.060	5954	5954	none

	Profile	Utilization Summary					Instance	
		BRAM	DSP	FF	LUT	URAM	fadd	fmul
	Available	280	220	106400	53200	0		
1.1	Baseline (AutoPipe)	13	5	1050	2000	0	1	1
1.1	Baseline (NoPipe)	14	5	817	1635	0	1	1
??	L3 Pipelining	14	5	921	1713	0	1	1
??	L2 Pipelining	13	10	24840	22620	0	2	2
??	L1 Pipelining (1WnR)	70	800	415044	243128	0	160	160
??	L1 Pipelining (T2P)	16	100	312992	120185	0	20	20

1. "{L1, L2, L3} Pipelining" are based on Baseline (NoPipe).
2. "L2/Partition" are based on L2 Pipelining.

3. the overall device utilization (as Total per Resource).
4. your measured system speedup over the fixed-point CPU implementation
5. your measured classification accuracy on the 8k MNIST test sample
6. how many multipliers are instantiated in your desing?
7. report the initiation interval of the matrix multiplication loop that you pipelined
8. given the number of multipliers in your design and input throughput via the AXI port, is the design bandwidth- or compute-limited?

3 Part 3: Open-ended design optimization (30 marks)

Vitis High-Level Synthesis User Guide