Custom Acceleration with FPGAs

CS5222 – Project 2

SHEN JIAMIN

A0209166A
`shen_jiamin@u.nus.edu`

Mar 31, 2022

Outline

Experiment Environment

- Ubuntu 20.04.4 LTS (GNU/Linux 5.4.0-105-generic x86_64)
- PYNQ v2.7.0
- Vivado 2020.2

Problem with Vivado HLS 2017.1

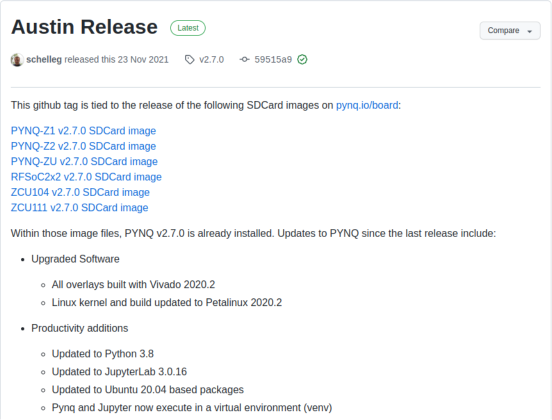
Due to the change in glibc,

- Vivado HLS 2017.1 doesn't work on Ubuntu 20.04 or Ubuntu 18.04
- It can only run on Ubuntu 16.04

```
$ vivado_hls -f hls.tcl
================================================================
  Vivado(TM) HLS - High-Level Synthesis from C, C++ and SystemC
  Version 2017.1
  Build 1846317 on Fri Apr 14 19:19:38 MDT 2017
  Copyright (C) 1986-2017 Xilinx, Inc. All Rights Reserved.
================================================================
...
/data/Xilinx/Vivado_HLS/2017.1/lnx64/tools/gcc/bin/../lib/gcc/x86_64-unknown-linux-gnu/4.6
↪  fatal error: xlocale.h: No such file or directory
compilation terminated.
make: *** [csim.mk:74: obj/mmult_test.o] Error 1
ERROR: [SIM 211-100] 'csim_design' failed: compilation error(s).
```

Lift the IDE Version

- Latest release of PYNQ is v2.7.0
- PYNQ v2.7.0 requires Vivado 2020.2



Changes for Vivado 2020.2

HLS Synthesis

- Vivado HLS is changed to Vitis HLS

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

- Pipelining enabled by default

```
config_compile -pipeline_loops 0
```

- RAM can be inferred to 1WnR

```
#pragma HLS bind_storage variable=in_buf type=RAM_T2P
```

- TLAST signal is not generated properly.

```
void mmult_hw(hls::stream<AXI_VAL> &in_stream, hls::stream<AXI_VAL>
↪ &out_stream);
```

Changes for Vivado 2020.2

System Synthesis

- Bypass version check in `classifier.tcl`

  ```
  set scripts_vivado_version 2020.2
  ```

- (Y2K22) Bad lexical cast: source type value could not be interpreted as target
  - `$ faketime -f "-1y" make`
  - Patch: `https://support.xilinx.com/s/article/76960`
- Fail to export `classifier.hdf`

  It doesn't matter, see next page

Changes for PYNQ v2.7.0

- Tcl parsing removed - please generate and use an HWH file for Overlays

```
$ scp `find -name "*.bit"` xilinx@ip-address:~/classifier.bit
$ scp `find -name "*.hwh"` xilinx@ip-address:~/classifier.hwh
```

- The DMA interface has changed.

```
ol = Overlay("/home/xilinx/classifier.bit")
ol.download()

dma_mm2s = ol.axi_dma_0
dma_s2mm = ol.axi_dma_1
```

- DMA buffer is be typed

```
output_buffer = allocate(shape=(BATCH * CLASSES,), dtype=np.float32)
c = np.reshape(np.array(output_buffer), (BATCH, CLASSES))
```

Outline

Experiment Environment

Result for Part 1 and Part 2

Design for Part 3

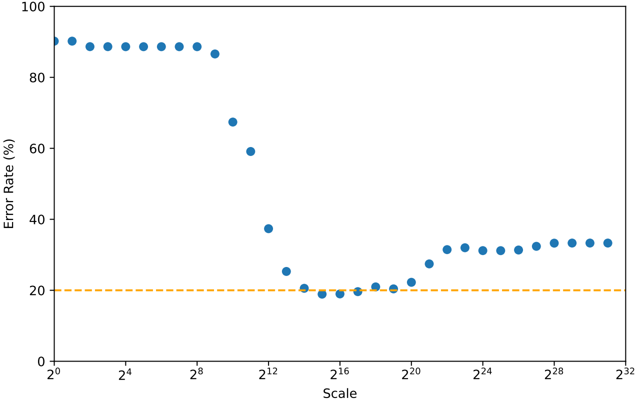
Result for Part 1

Profile		Latency		
		Overall	Normalized	
A	Baseline (NoPipe)	228022	28502.8	1.0x
B	L2 Pipelining (T2P)	13885	1735.6	16.4x
C	Partition (`dim`=2, `factor`=16)	4279	534.9	53.3x
D	Amortizing (`batch`=256)	57103	223.1	127.8x
E	Tiling (`batch`=2048, `tile`=128)	458106	223.7	127.4x
F	Hardware	949242	463.5	61.5x

On Hardare: Accuracy = 86.96%, FPGA Speedup = 4.83x

Result for Part 2

Pre-experiment



Result for Part 2

Experiment Result

- SCALE $= 2^{16}$
- Hardware Design
  - 127 multipliers on LUT, 129 multiply-accumulate operators on DSP
  - Overall Latency: 386378 cycles
  - Normalized Latency: 467.17 cycles (604.3x)
- Evaluation
  - Accuracy: 80.04%
  - FPGA Speedup: 34.94x

Outline

Optimization against Latency

Enhancing a Single AXI Transfer

Expanding `TDATA` width improves the rate of transfer.

- Original design transfers 64 bits per cycle
  - A tile of 128 inputs has $128 \times 16^2 \times 8 = 2^{18}$ bits
  - It needs $2^{12} = 4096$ cycles to transfer
- Expand the AXI `TDATA` bus to 256 bits

`TDATA` width (bits)	Latency (cycles)	Trip Count
64	4096	128
128	2048	128
256	1024	128

Optimization against Latency

Reducing the Input Dimension

Reducing input dimension lowers the number of bits to transfer (and also simplifies the computation).

- Input dimension reduced to 8×8
- Input depth reduced to 4 bits

But that makes the prediction accuracy declined to 82.96%.

Optimization against Accuracy

New classifier architecture:

- Input layer: 64×16 8-bit integers
- Activation layer: ReLU
- Output layer: 16×10 8-bit integers with a bias of 10 16-bit integers.

Prediction accuracy:

- Floating-point numbers: 89.88%
- Fixed-point numbers: 88.92%

IP Synthesis

Latency Estimates

Overall Latency: 159646 cycles

Loop	Latency	Iter Latency	Trip Count	Pipelined
- `LOAD_OFFSET`	10	1	10	yes
- `LOAD_WEIGHT1`	32	2	16	yes
- `LOAD_WEIGHT2`	40	9	5	yes
- `LT`	159552	2493	64	no
+ `LOAD_INPUT`	128	1	128	yes
+ `COMPUTE_INPUT`	2053	7	2048	yes
+ `COMPUTE_OUTPUT`	134	8	128	yes
+ `STORE_OUTPUT`	129	3	128	yes

IP Synthesis

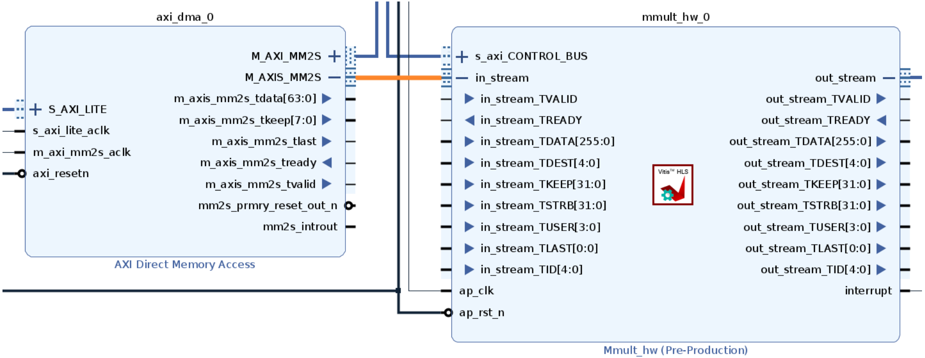
Utilization Estimates

Implementation	Module	Count
LUT	`mul_8s_4ns_12_1_1`	32
DSP	`mul_mul_16s_8s_16_4_1`	70
DSP	`mac_muladd_16s_8s_16ns_16_4_1`	10
DSP	`mac_muladd_16s_8s_16s_16_4_1`	80
DSP	`mac_muladd_8s_4ns_12s_13_4_1`	32

Name	BRAM_18K	DSP	FF	LUT
Total	202	192	6620	9763
Utilization (%)	72	87	6	18

System Synthesis

Bus Interface property `TDATA_NUM_BYTES` does not match



Evaluation

- Latency
  - FPGA: 3.85 ms
  - CPU: 169.12 ms
  - Speedup: 43.97x
- Accuracy
  - 88.18%