Update on Ara

01/09/2021

Matteo Perotti
Matheus Cavalcante
Nils Wistoff

Professor Luca Benini
Integrated Systems Laboratory
ETH Zürich

Summary

- Fix the cycle count
- Optimize convolution 3x3
  - Bug fixes
  - Insights
  - Optimization
- Optimize convolution 7x7

Improve the Cycle Count

```
start = start_timer();

vectorial_kernel();

end = end_timer();
cycle_count = end - start;
```

```
vectorial_kernel() {
  ...
  vload
  vmul
  vstore (in processing)
}
```


Problem! Maybe Ara is still computing vector instructions when end_timer() is called!

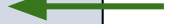
Improve the Cycle Count

```
start = start_timer();

vectorial_kernel();

fence;
end = end_timer();
cycle_count = end - start;
```

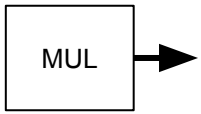
```
vectorial_kernel() {
  ...
  vload
  vmul
  vstore (will complete)
}
```



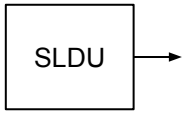
fence: wait for the vector store to complete

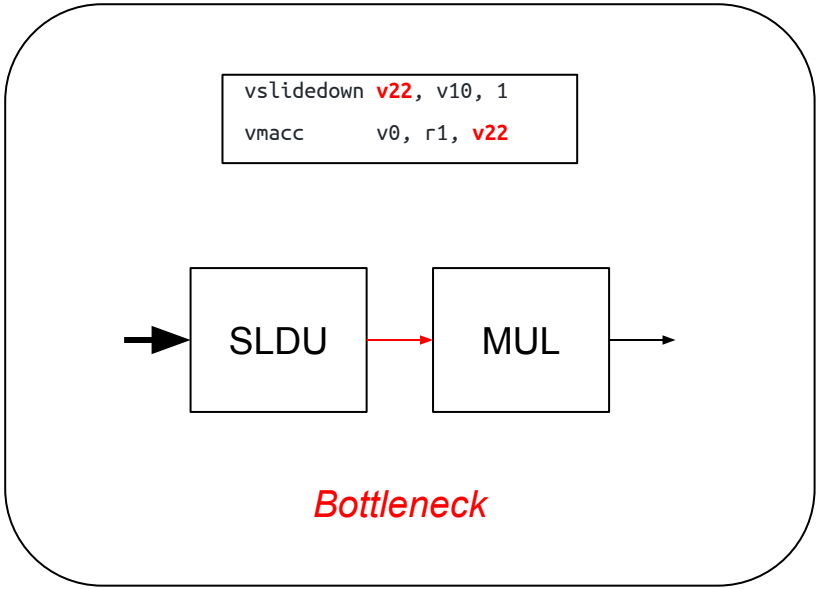
Throughput of the Units

- Full Throughput Units:
  N_LANES * 8 B/cycle
  - FPU
  - Multiplier
  - ALU



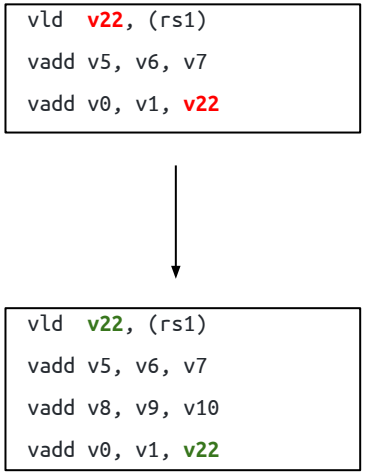
- Half Throughput Units:
  N_LANES * 4 B/cycle
  - Load/Store Unit
  - Slide Unit





Hazard checks: RAW

- Theoretically: full throughput
- Practically: the hazard check mechanism slows down the second vadd
- Solution: change the instruction flow



Hazard checks: WAW

- No real hazard if the functional unit is the same
- Practically: it is considered as a hazard
- Issue: the stalls on the first instruction writes stall the second instruction reads

Real WAW hazard:

```
vmul v5, v6, v7
vadd v5, v1, v22
```

Fake WAW hazard:

```
vadd v5, v6, v7
vadd v5, v1, v22
```

Fix: AXI Compliancy

```
vst v0, (r5)
vld v1, (r6)
```

Before the fix:

1. Start write transaction on AW
2. Start read transaction on AR
3. Write on the W channel

POSSIBLE DEADLOCK

4. Read from the R channel

Fix: AXI Compliancy

```
vst v0, (r5)
vld v1, (r6)
```

Before the fix:

1. Start write transaction on AW
2. Start read transaction on AR
3. Write on the W channel
4. Read from the R channel

In this case, AXI does not specify an ordering of the responses on opposite channels.

When Ara wants to write, it is not ready to read.

Solution: complete the first transaction before starting the second one.

Fix: AXI Compliancy

```
vst v0, (r5)
vld v1, (r6)
```

Before the fix:

1. Start write transaction on AW
2. Start read transaction on AR
3. Write on the W channel
4. Read from the R channel

After the fix:

1. Start write transaction on AW
2. Write on the W channel
3. Start read transaction on AR
4. Read from the R channel

Fix: AXI Compliancy

```
vst v0, (r5)
vld v1, (r6)
```

Before the fix:

1. Start write transaction on AW
2. Start read transaction on AR
3. Write on the W channel
4. Read from the R channel

After the fix:

1. Start write transaction on AW
2. Write on the W channel
3. Start read transaction on AR
4. Read from the R channel

↓

Okay since the memory is single-ported

Instruction dispatching mechanism

- The main sequencer dispatches instructions to the functional units (FUs)
- If at least one FU is not ready, no instruction is dispatched, regardless of the instruction
- If one FU is full of instructions, it is no more ready to accept new ones
- Sizing the instruction queues accordingly is crucial to avoid stalls

Vector stores - Scalar loads

- Ariane does not execute loads if there is an ongoing vector store
- This happens to enforce coherency

Performance

Integer convolution:

- 112x112 input matrix
- 3x3 filter convolution
- Utilization: 89.8%

Floating-Point convolution:

- 112x112 input matrix
- 3x3 filter convolution
- Utilization: 89.4%

Ongoing: optimization of the 7x7 convolution, to make a comparison with AraV1