RISC-V Processor Design 🚀

Building Tiny Vedas

Marco Spaziani Brunella, PhD

Lecture 9

Agenda

- DCCM
- Aligned loads
- Unaligned loads and performance improvements
- Stores and performance improvements

Data Closely Coupled Memory (DCCM)

- Similar to ICCM
- 4K, 32-bit words x 1024 rows
- Byte addressable
- One read and one write port
- 1cc latency for loads
- In-order responses

Aligned loads

- Loads are aligned to 4-byte boundaries
- Unaligned = we need to load two rows = two reads = slower
- Which loads are aligned?
    - Byte loads are always aligned
    - Half-word loads are aligned if the address is even
    - Word loads are aligned if the address is a multiple of 4

Unaligned loads Example

```
+---+---+---+---+
| 0 | 1 | 2 | 3 | 0x00
+---+---+---+---+
| 4 | 5 | 6 | 7 | 0x04
+---+---+---+---+
| 8 | 9 | A | B | 0x08
+---+---+---+---+
| C | D | E | F | 0x0C
+---+---+---+---+

lw x1, 2(x0) # x1 = B2|B3|B4|B5
lh x2, 3(x0) # x2 = B3|B4
```

How we build a load and store unit for the current DCCM architecture?

- We need to arbitrate access to the only read port
- If the load is aligned, we just need to read one row and shift the data right (if needed)
- If the load is unaligned:
    - Stall the pipeline (to send a bubble in)
    - Read the first row in DC1 stage
    - Read the second row in DC2 stage
    - Combine the results in DC3 stage and send to write back

How can we improve the performance of unaligned loads?

- No other solution than get a DCCM with an extra read port

Stores

- Here the concept of alignement is still important
- But the other important thing is "are we writing an entire row?"
- If not, we need to read the row first, update the data, and then write the row back
  - The only case where we DO NOT need to read the row first is when we are storing an aligned word

Performance optimization: store forwarding

- If we are writing to an address that is being read (either by a load or another store that needs loads), we can forward the data
- This is a very powerful optimization

How to avoid loads when storing?

- Again, this falls back to the memory architecture and bus used
- We tipically get rid of the loads on writes by using byte enables
    - More complex bus architectures, like AXI and AHB are designed for this (and much more)