

Update on Ara

26/05/2021

Matteo Perotti

Matheus Cavalcante

Nils Wistoff

Professor Luca Benini

Integrated Systems Laboratory

ETH Zürich

Ongoing work

- **Benchmarks + Verification**
 - Jacobi2d now works
 - Fixed:
 - Misaligned stores
 - Misc bugs
 - Added:
 - Strided memory operations
 - New riscv-tests
 - Misaligned mem ops
 - Strided mem ops
 - vsetivli (config)

- **TO**

Aligned stores

Support for aligned stores

- `vuint64m1_t` `ara_vec` = {0xb4092d9a981142fa, 0xd7548e553400af32,
0xa0a0a0a0a0a0a0a0, 0x3333333333333333,
0x4444444444444444, 0x5555555555555555};

Support for aligned stores

- `vuint64m1_t` `ara_vec` = {0xb4092d9a981142fa, 0xd7548e553400af32,
0xa0a0a0a0a0a0a0a0, 0x3333333333333333,
0x4444444444444444, 0x5555555555555555};



VRF

Support for aligned stores

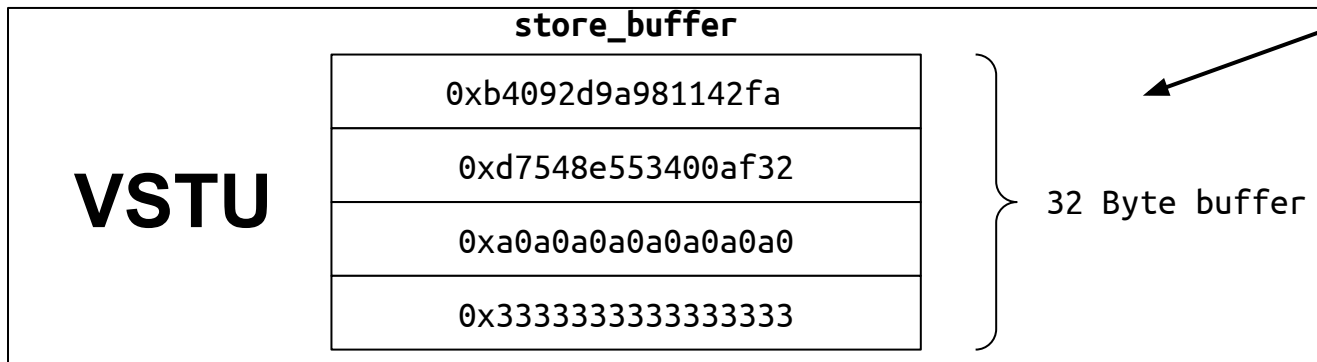
- `vuint64m1_t ara_vec = {0xb4092d9a981142fa, 0xd7548e553400af32,
0xa0a0a0a0a0a0a0a0, 0x3333333333333333,
0x4444444444444444, 0x5555555555555555};`
- `vse64_v_u64m1(&mem_buf[0], ara_vec, vl); // vl == 6 elements`



VRF

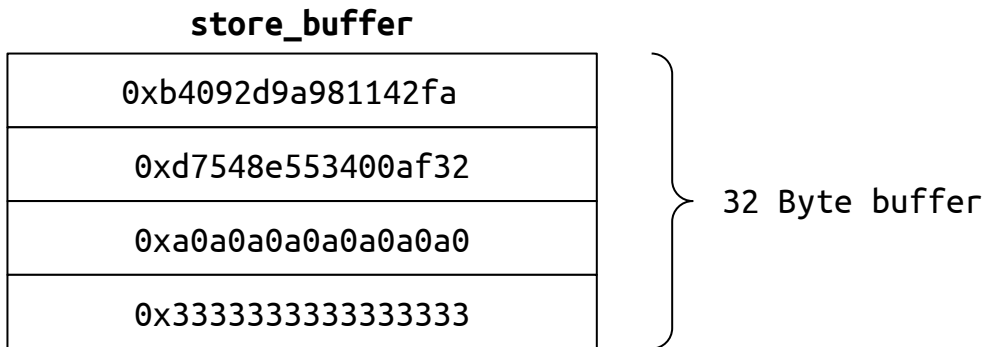
Support for aligned stores

- `vuint64m1_t ara_vec = {0xb4092d9a981142fa, 0xd7548e553400af32, 0xa0a0a0a0a0a0a0a0, 0x3333333333333333, 0x4444444444444444, 0x5555555555555555};`
- `vse64_v_u64m1(&mem_buf[0], ara_vec, vl); // vl == 6 elements`



Support for aligned stores

- `vuint64m1_t ara_vec = {0xb4092d9a981142fa, 0xd7548e553400af32,
0xa0a0a0a0a0a0a0a0, 0x3333333333333333,
0x4444444444444444, 0x5555555555555555};`
- `vse64_v_u64m1(&mem_buf[0], ara_vec, vl); // vl == 6 elements`

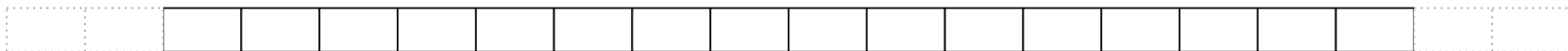


Support for aligned stores

- `vuint64m1_t ara_vec = {0xb4092d9a981142fa, 0xd7548e553400af32, 0xa0a0a0a0a0a0a0a0, 0x3333333333333333, 0x4444444444444444, 0x5555555555555555};`
- `vse64_v_u64m1(&mem_buf[0], ara_vec, vl); // vl == 6 elements`

`mem_buf[0]`

16 Byte AXI Data Bus



0x100

0x110

store_buffer

0xb4092d9a981142fa
0xd7548e553400af32
0xa0a0a0a0a0a0a0a0
0x3333333333333333

32 Byte buffer

Support for aligned stores

- `vuint64m1_t ara_vec = {0xb4092d9a981142fa, 0xd7548e553400af32,
0xa0a0a0a0a0a0a0a0, 0x3333333333333333,
0x4444444444444444, 0x5555555555555555};`
- `vse64_v_u64m1(&mem_buf[0], ara_vec, vl); // vl == 6 elements`

mem_buf[0]

16 Byte AXI Data Bus

		0xfa	0x42	0x11	0x98	0x9a	0x2d	0x09	0xb4	0x32	0xaf	0x00	0x34	0x55	0x8e	0x54	0xd7		
--	--	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	--	--

0x100

0x110

store_buffer

0xb4092d9a981142fa
0xd7548e553400af32
0xa0a0a0a0a0a0a0a0
0x3333333333333333

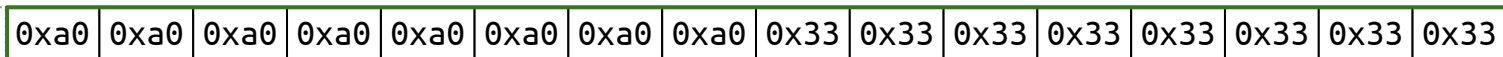
32 Byte buffer

Support for aligned stores

- `vuint64m1_t ara_vec = {0xb4092d9a981142fa, 0xd7548e553400af32, 0xa0a0a0a0a0a0a0a0, 0x3333333333333333, 0x4444444444444444, 0x5555555555555555};`
- `vse64_v_u64m1(&mem_buf[0], ara_vec, vl); // vl == 6 elements`

mem_buf[2]

16 Byte AXI Data Bus



0x110

0x120

store_buffer

0xb4092d9a981142fa
0xd7548e553400af32
0xa0a0a0a0a0a0a0a0
0x3333333333333333

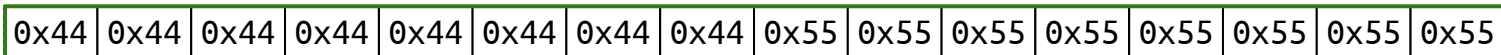
32 Byte buffer

Support for aligned stores

- `vuint64m1_t ara_vec = {0xb4092d9a981142fa, 0xd7548e553400af32,
0xa0a0a0a0a0a0a0a0, 0x3333333333333333,
0x4444444444444444, 0x5555555555555555};`
- `vse64_v_u64m1(&mem_buf[0], ara_vec, vl); // vl == 4 elements`

mem_buf[2]

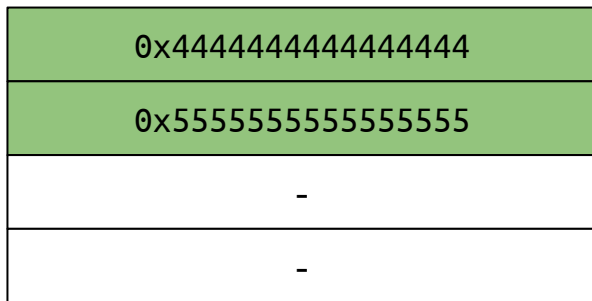
16 Byte AXI Data Bus



0x120

0x130

store_buffer



32 Byte buffer

Misaligned stores

Support for misaligned stores

- `vuint64m1_t` ara_vec = {0xb4092d9a981142fa, 0xd7548e553400af32,
0xa0a0a0a0a0a0a0a0, 0x3333333333333333,
0x4444444444444444, 0x5555555555555555};

Support for misaligned stores

- `vuint64m1_t ara_vec = {0xb4092d9a981142fa, 0xd7548e553400af32,
0xa0a0a0a0a0a0a0a0, 0x3333333333333333,
0x4444444444444444, 0x5555555555555555};`



VRF

Support for misaligned stores

- `vuint64m1_t ara_vec = {0xb4092d9a981142fa, 0xd7548e553400af32,
0xa0a0a0a0a0a0a0a0, 0x3333333333333333,
0x4444444444444444, 0x5555555555555555};`
- `vse64_v_u64m1(&mem_buf[1], ara_vec, vl); // vl == 6 elements`



VRF

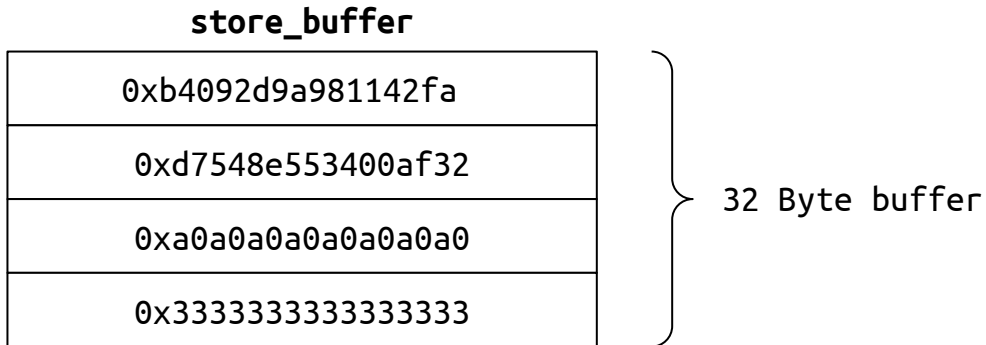
Support for misaligned stores

- `vuint64m1_t ara_vec = {0xb4092d9a981142fa, 0xd7548e553400af32, 0xa0a0a0a0a0a0a0a0, 0x3333333333333333, 0x4444444444444444, 0x5555555555555555};`
- `vse64_v_u64m1(&mem_buf[1], ara_vec, vl); // vl == 6 elements`



Support for misaligned stores

- `vuint64m1_t ara_vec = {0xb4092d9a981142fa, 0xd7548e553400af32,
0xa0a0a0a0a0a0a0a0, 0x3333333333333333,
0x4444444444444444, 0x5555555555555555};`
- `vse64_v_u64m1(&mem_buf[1], ara_vec, vl); // vl == 6 elements`

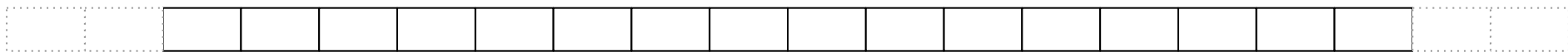


Support for misaligned stores

- `vuint64m1_t ara_vec = {0xb4092d9a981142fa, 0xd7548e553400af32, 0xa0a0a0a0a0a0a0a0, 0x3333333333333333, 0x4444444444444444, 0x5555555555555555};`
- `vse64_v_u64m1(&mem_buf[1], ara_vec, vl); // vl == 6 elements`

mem_buf[0]

16 Byte AXI Data Bus



0x100

0x110

store_buffer

0xb4092d9a981142fa
0xd7548e553400af32
0xa0a0a0a0a0a0a0a0
0x3333333333333333

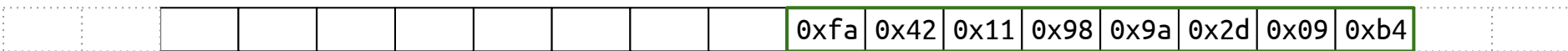
32 Byte buffer

Support for misaligned stores

- `vuint64m1_t ara_vec = {0xb4092d9a981142fa, 0xd7548e553400af32, 0xa0a0a0a0a0a0a0a0, 0x3333333333333333, 0x4444444444444444, 0x5555555555555555};`
- `vse64_v_u64m1(&mem_buf[1], ara_vec, vl); // vl == 6 elements`

mem_buf[0]

16 Byte AXI Data Bus



0x100

0x110

store_buffer

0xb4092d9a981142fa
0xd7548e553400af32
0xa0a0a0a0a0a0a0a0
0x3333333333333333

32 Byte buffer

Support for misaligned stores

- `vuint64m1_t ara_vec = {0xb4092d9a981142fa, 0xd7548e553400af32, 0xa0a0a0a0a0a0a0a0, 0x3333333333333333, 0x4444444444444444, 0x5555555555555555};`
- `vse64_v_u64m1(&mem_buf[1], ara_vec, vl); // vl == 6 elements`

mem_buf[2]

16 Byte AXI Data Bus

		0x32	0xaf	0x00	0x34	0x55	0x8e	0x54	0xd7	0xa0	0xa0	0xa0	0xa0	0xa0	0xa0	0xa0		
--	--	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	--	--

0x110

0x120

store_buffer

0xb4092d9a981142fa
0xd7548e553400af32
0xa0a0a0a0a0a0a0a0
0x3333333333333333

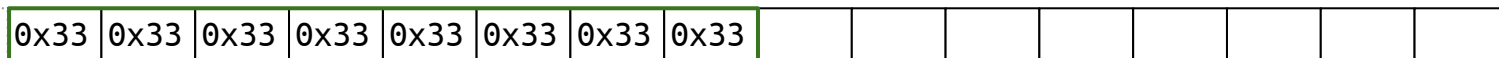
32 Byte buffer

Support for misaligned stores

- `vuint64m1_t ara_vec = {0xb4092d9a981142fa, 0xd7548e553400af32,
0xa0a0a0a0a0a0a0a0, 0x3333333333333333,
0x4444444444444444, 0x5555555555555555};`
- `vse64_v_u64m1(&mem_buf[1], ara_vec, vl); // vl == 6 elements`

mem_buf[2]

16 Byte AXI Data Bus



0x120

0x130

We need the next element to complete the AXI beat. But if we get the next element, we lose the previous one.

store_buffer

0xb4092d9a981142fa
0xd7548e553400af32
0xa0a0a0a0a0a0a0a0
0x3333333333333333

32 Byte buffer

Support for misaligned stores

Multiple solutions:

- **Double buffer**
 - We always keep also the next elements
- **Independent ready-signals to the lanes**
 - Each buffer entry can be updated independently
- **Reduce the effective AXI width in case of misaligned store**
 - Less intrusive solution
 - Faster to implement
 - Misaligned stores are not common

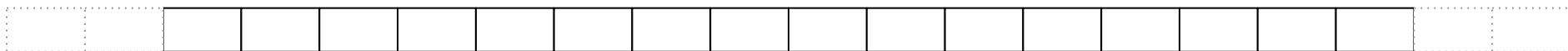


Support for misaligned stores

- `vuint64m1_t ara_vec = {0xb4092d9a981142fa, 0xd7548e553400af32, 0xa0a0a0a0a0a0a0a0, 0x3333333333333333, 0x4444444444444444, 0x5555555555555555};`
- `vse64_v_u64m1(&mem_buf[1], ara_vec, vl); // vl == 6 elements`

mem_buf[0]

8 Byte effective AXI Data Bus



0x100

0x110

store_buffer

0xb4092d9a981142fa
0xd7548e553400af32
0xa0a0a0a0a0a0a0a0
0x3333333333333333

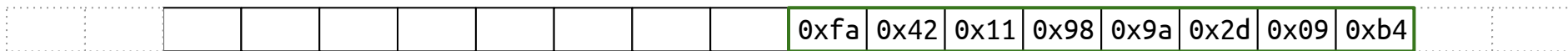
32 Byte buffer

Support for misaligned stores

- `vuint64m1_t ara_vec = {0xb4092d9a981142fa, 0xd7548e553400af32, 0xa0a0a0a0a0a0a0a0, 0x3333333333333333, 0x4444444444444444, 0x5555555555555555};`
- `vse64_v_u64m1(&mem_buf[1], ara_vec, vl); // vl == 6 elements`

mem_buf[0]

8 Byte effective AXI Data Bus



0x100

0x110

store_buffer

0xb4092d9a981142fa
0xd7548e553400af32
0xa0a0a0a0a0a0a0a0
0x3333333333333333

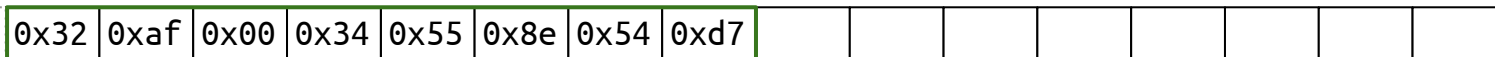
32 Byte buffer

Support for misaligned stores

- `vuint64m1_t ara_vec = {0xb4092d9a981142fa, 0xd7548e553400af32, 0xa0a0a0a0a0a0a0a0, 0x3333333333333333, 0x4444444444444444, 0x5555555555555555};`
- `vse64_v_u64m1(&mem_buf[1], ara_vec, vl); // vl == 6 elements`

mem_buf[2]

8 Byte effective AXI Data Bus



0x110

0x120

store_buffer

0xb4092d9a981142fa
0xd7548e553400af32
0xa0a0a0a0a0a0a0a0
0x3333333333333333

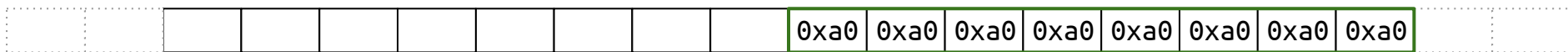
32 Byte buffer

Support for misaligned stores

- `vuint64m1_t ara_vec = {0xb4092d9a981142fa, 0xd7548e553400af32,
0xa0a0a0a0a0a0a0a0, 0x3333333333333333,
0x4444444444444444, 0x5555555555555555};`
- `vse64_v_u64m1(&mem_buf[1], ara_vec, vl); // vl == 6 elements`

mem_buf[2]

8 Byte effective AXI Data Bus



0x110

0x120

store_buffer

0xb4092d9a981142fa
0xd7548e553400af32
0xa0a0a0a0a0a0a0a0
0x3333333333333333

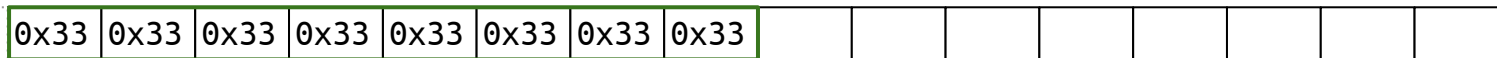
32 Byte buffer

Support for misaligned stores

- `vuint64m1_t ara_vec = {0xb4092d9a981142fa, 0xd7548e553400af32,
0xa0a0a0a0a0a0a0a0, 0x3333333333333333,
0x4444444444444444, 0x5555555555555555};`
- `vse64_v_u64m1(&mem_buf[1], ara_vec, vl); // vl == 6 elements`

mem_buf[2]

8 Byte effective AXI Data Bus



0x120

0x130

store_buffer

0xb4092d9a981142fa
0xd7548e553400af32
0xa0a0a0a0a0a0a0a0
0x3333333333333333

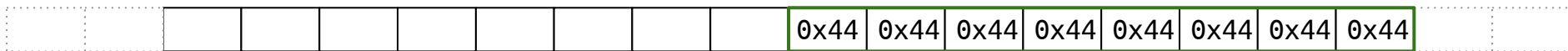
32 Byte buffer

Support for misaligned stores

- `vuint64m1_t ara_vec = {0xb4092d9a981142fa, 0xd7548e553400af32,
0xa0a0a0a0a0a0a0a0, 0x3333333333333333,
0x4444444444444444, 0x5555555555555555};`
- `vse64_v_u64m1(&mem_buf[1], ara_vec, vl); // vl == 6 elements`

mem_buf[2]

8 Byte effective AXI Data Bus



0x120

0x130

store_buffer

0x4444444444444444
0x5555555555555555
-
-

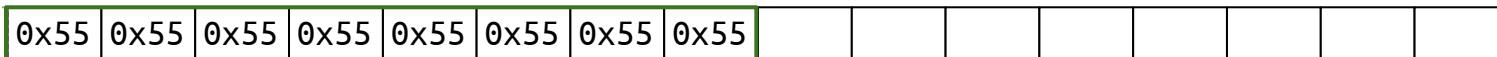
32 Byte buffer

Support for misaligned stores

- `vuint64m1_t ara_vec = {0xb4092d9a981142fa, 0xd7548e553400af32,
0xa0a0a0a0a0a0a0a0, 0x3333333333333333,
0x4444444444444444, 0x5555555555555555};`
- `vse64_v_u64m1(&mem_buf[1], ara_vec, vl); // vl == 6 elements`

mem_buf[2]

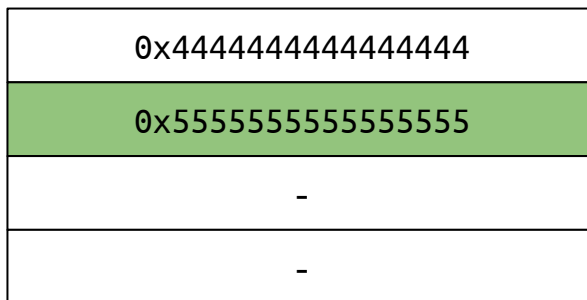
8 Byte effective AXI Data Bus



0x130

0x140

store_buffer



32 Byte buffer