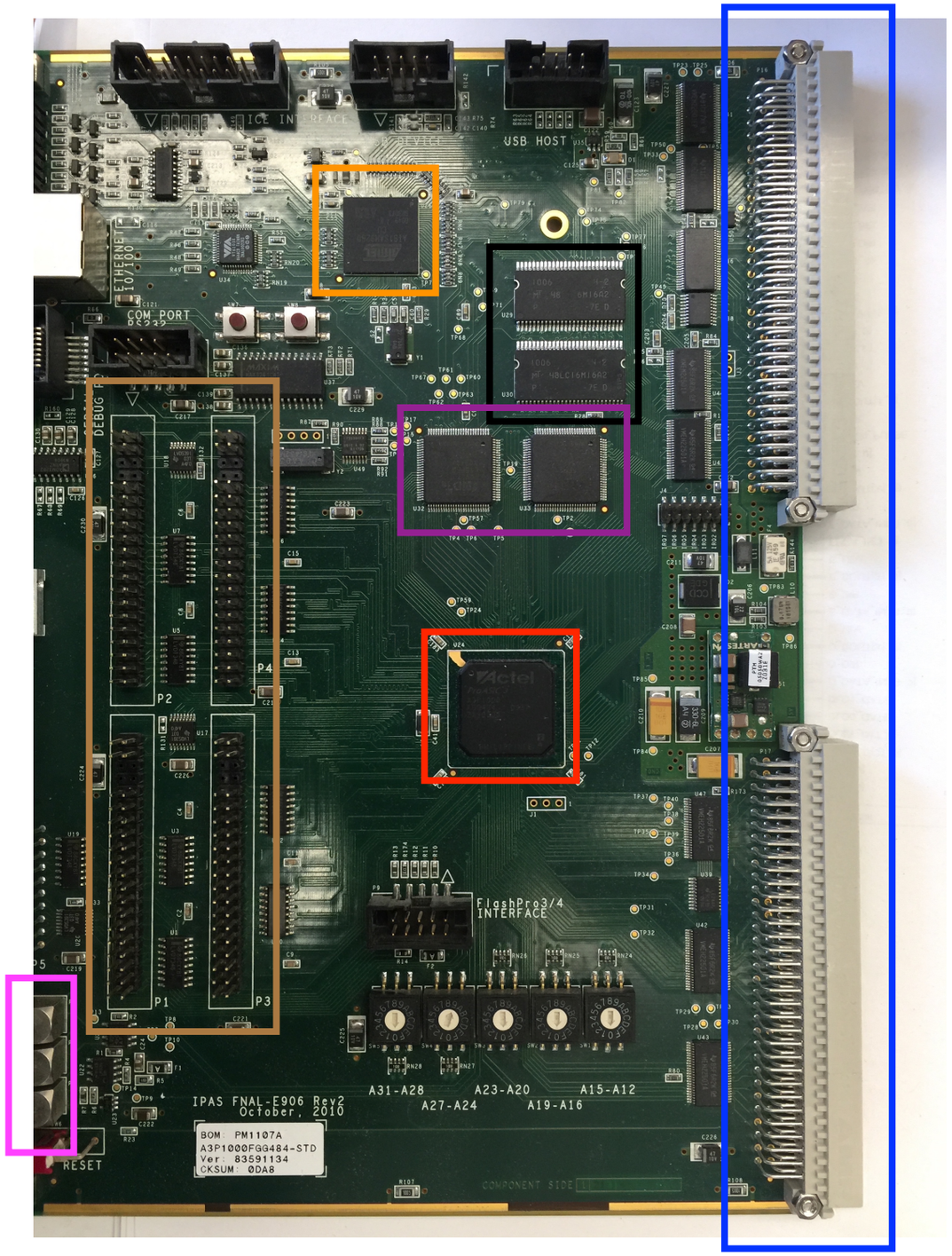
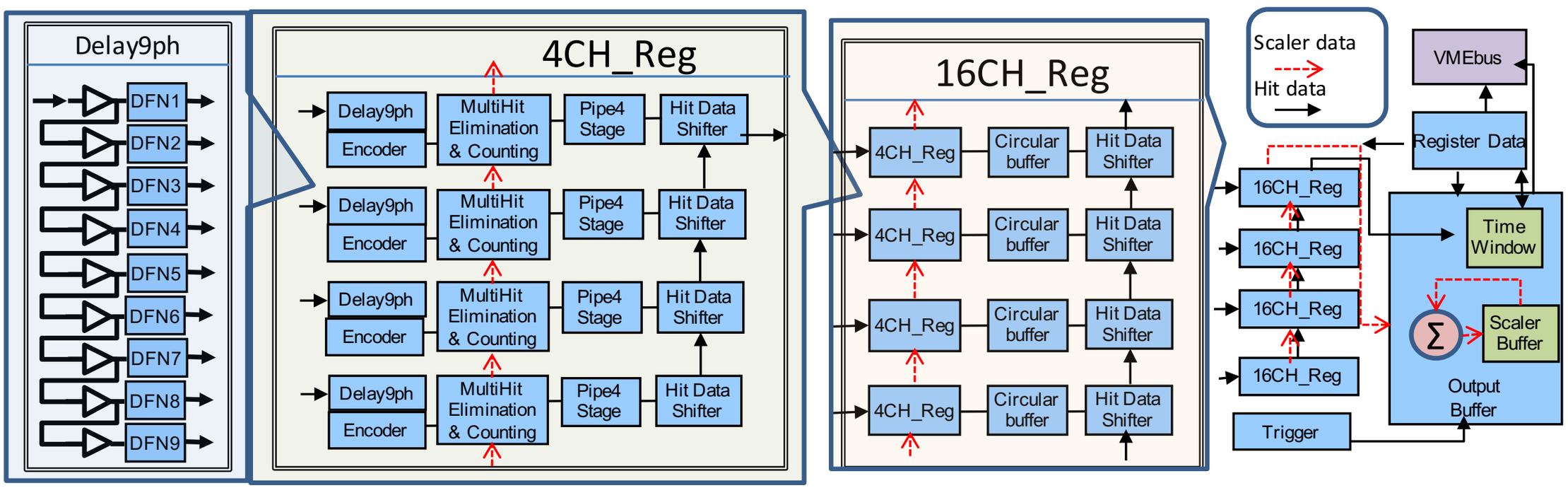
Key features of TW-TDC



- Standard VME interface
- 64-bit ECL inputs
- Two NIM inputs
- 1M Gates Actel ProASIC3 FPGA
- Microsemi AT91SAM9260 ARM processor
- 32K x 32 bit Dual-port SRAM
- 64MB SDRAM

Los Alamos NATIONAL LABORATORY EST. 1943

TDC readout chain and bottleneck

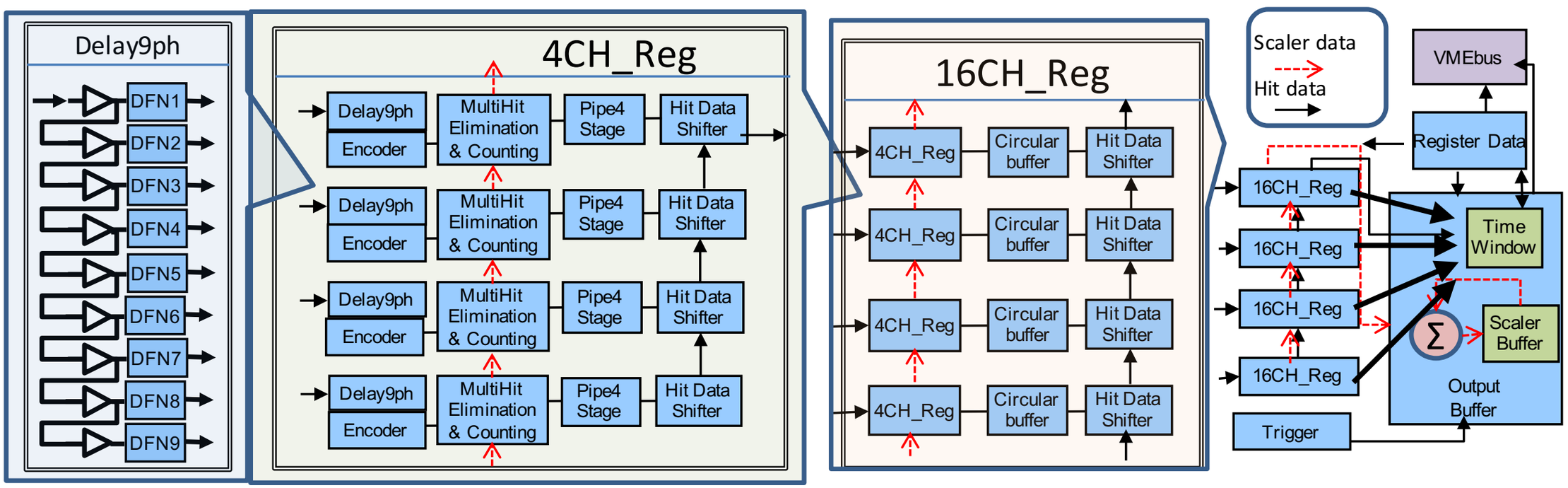


- Readout is controlled by 62.5 MHz clock
- After a valid hit gets digitized, the coarse and fine time are temporarily stored in a 4-hit buffer, and readout once every 64 ns
- Data from a group of 4 channels are saved into one of the 2 (or 4, or 8) circular buffer
- When trigger arrives, data in current circular buffer is copied to the event buffer (copy-in-progress time)
- The event buffer is mapped to the VME address space, which is readout by CPU sequentially (~4 hits per us)

Source of copy-in-progress time

Nbufs	Buffer size	Timing Window	CIP time
8	32	512 ns	32x16x16ns = 8 us
4	64	1024 ns	64x16x16ns = 16 us
2	128	2048 ns	128x16x16ns = 32 us

Reduce CIP time of TDC



Option 1: Split the event buffer into 4 blocks, move each 16-ch register block directly to the event buffer

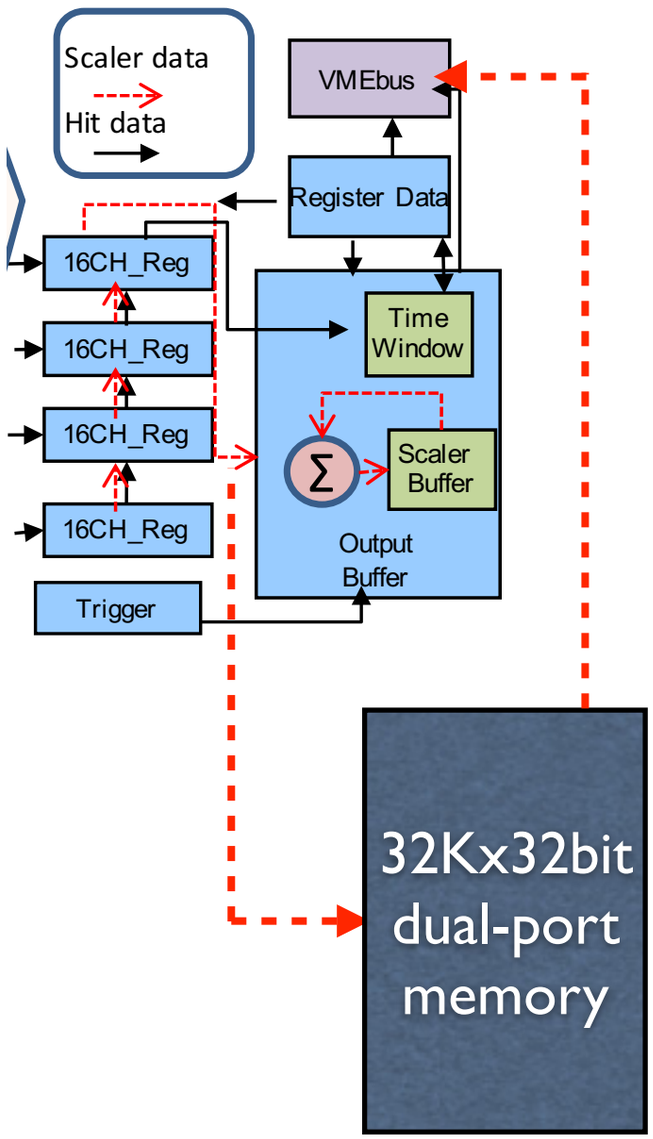
- **Pros:** 4 16-ch groups are transferred simultaneously, reduce CIP by 4x
- **Cons:** data is not continuous in event buffer, takes 4 instead of 1 DMA transfer to readout through VME bus
- **Caveat:** dual-port memory cannot operate this fast, may need to slow down the readout clock by 2x

Option 2: take advantage of the 2x2048 us circular buffer to save 2 events at the same time

- **Pros:** instead of dead for x us per event, we are now dead for x us per 2 events (only helpful when events comes in bursts)
- **Cons:** need to modify trigger supervisor behavior to delay TIR instruction once every 2 events for 2x us

Grass and Jin-Yuan are aiming at 10 us for now

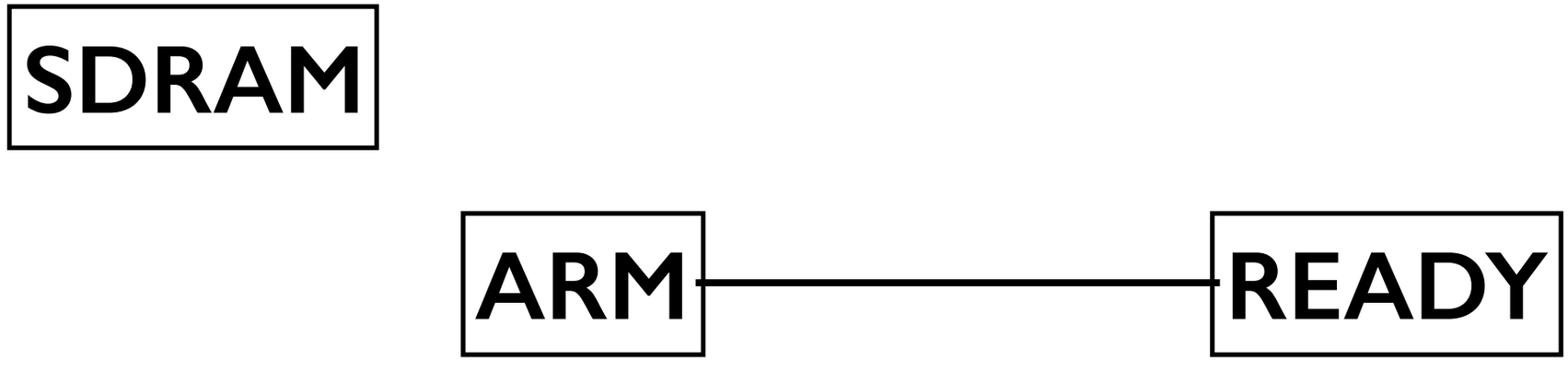
Eliminate data-transfer time



- Instead of going from circular buffer to event buffer, data is now also directed to dual-port memory. The same CIP time is needed.
- Instead of reading event buffer via VME bus (250 ns per hit), events in dual-port memory is read by ARM and saved in SDRAM (40 ns per hit)
- Dual-port memory is divided to 16 banks, so that we don't need to wait for ARM reading.
- Similar to event buffer, the dual-port memory is mapped to VME bus as well, so that we could read the data back when at leisure.

Los Alamos NATIONAL LABORATORY EST. 1943

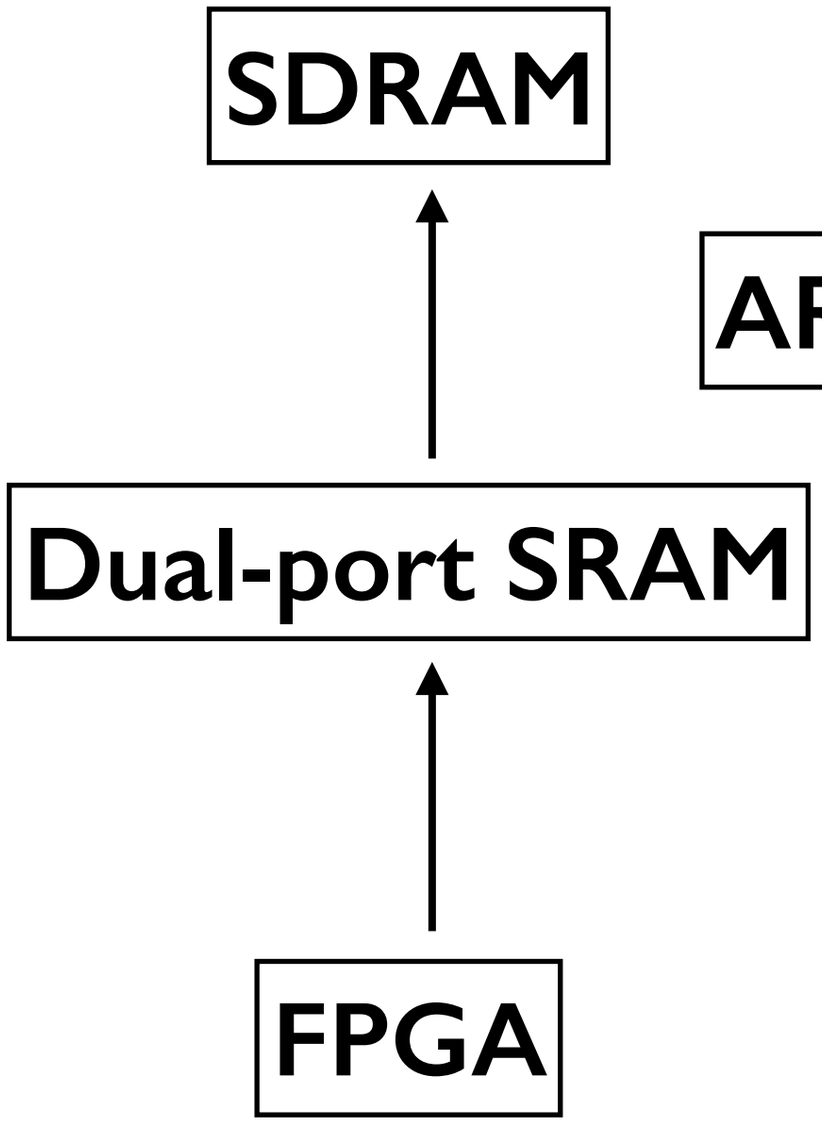
Buffering mechanism controlled by ARM



- Reset all the registers, and move to **BEAMON** state

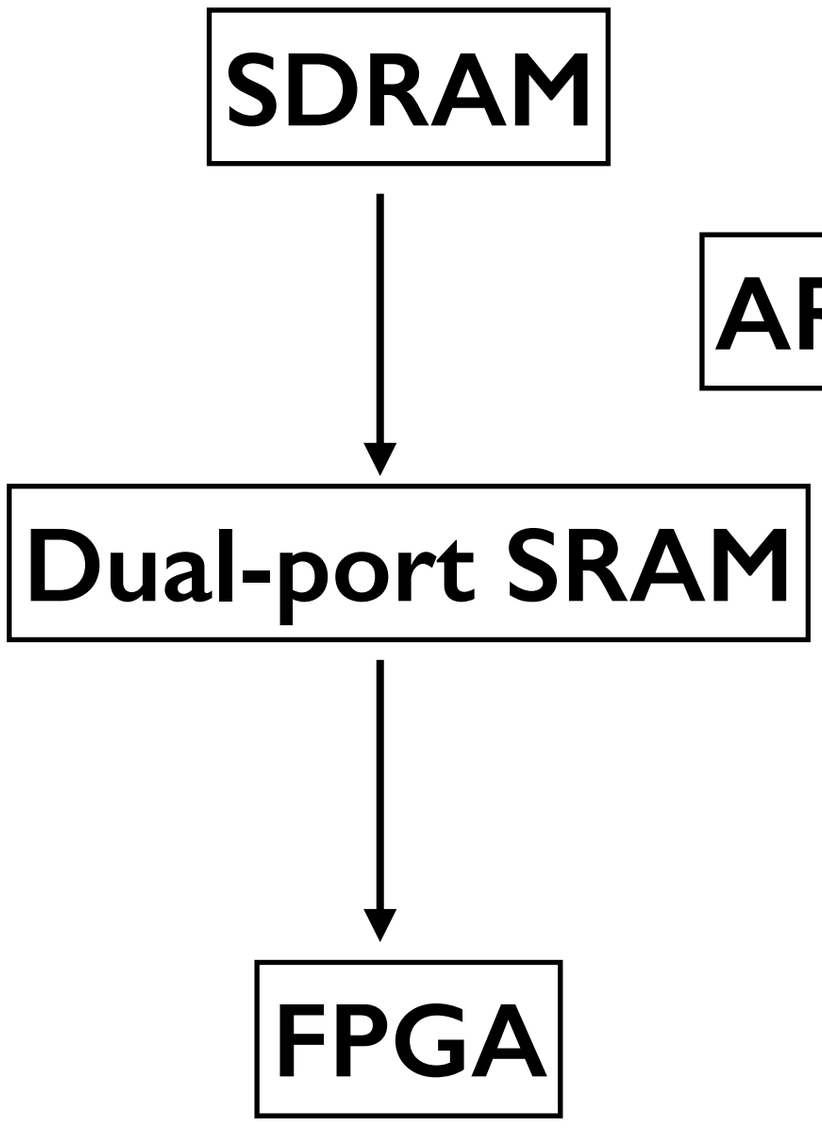
Los Alamos NATIONAL LABORATORY EST. 1943

Buffering mechanism controlled by ARM



- ARM keeps querying the header of each dual-port memory bank.
- If a valid header is seen:
  - read all the data and save them in SDRAM
  - set the header back to 0
- Upon receiving EOS event, FPGA sends interrupt to ARM, ARM enters **BEAMOFF** state

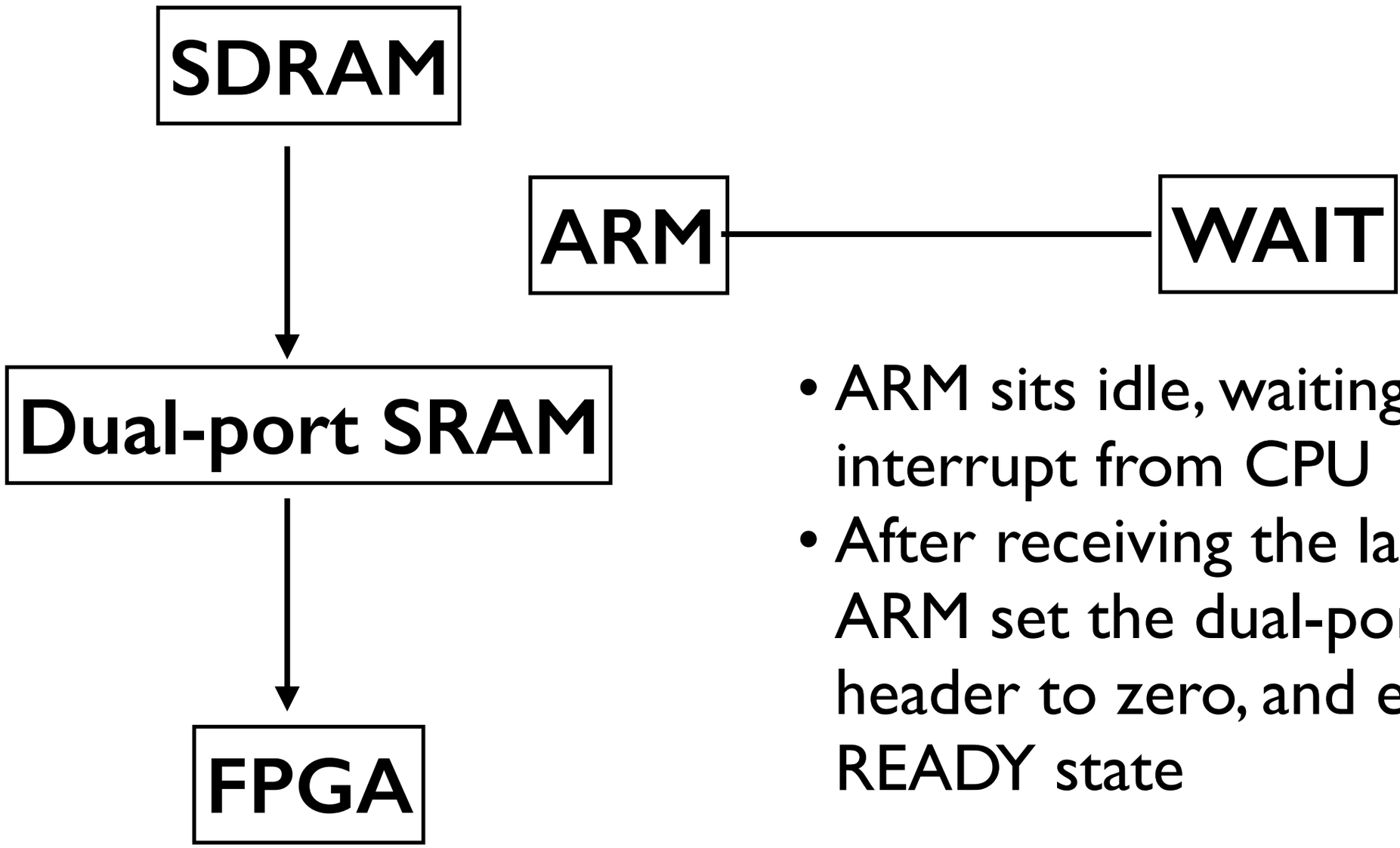
Buffering mechanism controlled by ARM



- ARM transfer a certain amount of data from SDRAM to Dual-port SRAM, wait for CPU to read
- After CPU finished reading on batch, an interrupt is sent to ARM to initiate next batch transfer
- When SDRAM is empty, ARM enters **WAIT** state

Los Alamos NATIONAL LABORATORY EST. 1943

Buffering mechanism controlled by ARM



- ARM sits idle, waiting for one last interrupt from CPU
- After receiving the last interrupt, ARM set the dual-port memory header to zero, and enters READY state

Los Alamos NATIONAL LABORATORY EST. 1943

Integration with CODA

- The DAQ limit on number of events taken per spill is ~420k.
- By design, one TDC can have no more than 256 hits per event, in read data taking we typically have no more than 64 hits in the most messy events, on average it's 20-30;
- One hit takes 4 bytes in memory.
- 64MB SDRAM can accommodate 65536 events at the worst case scenario, for real data taking, SDRAM is large enough for 260k - 520k events

- Dual-port memory is 128KB, thus it takes no more than 512 transfers to move everything from SDRAM to dual-port memory. It takes ~10 ms to read entire dual-port memory through VME bus
- During the beam on time, CODA keeps incrementing event counter, and write eventID to each TDC, and save an empty event with only the event type.
- EOS event will trigger the ARM to write back to dual-port memory. After that, flush event coming at 50Hz (20 ms apart) or so will be triggering the transfer
- Decoder will be responsible to re-assemble all the events