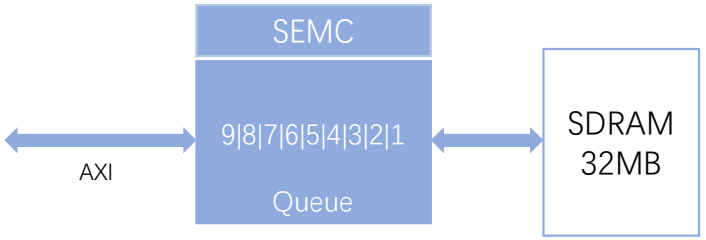
BMCR register

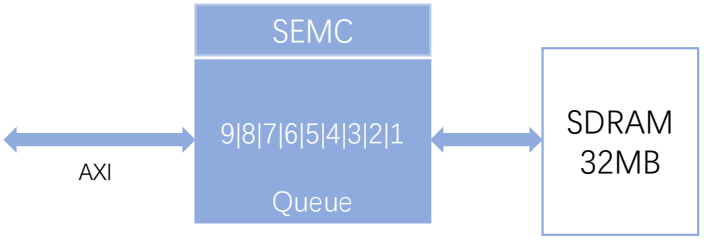
8h	Bus (AXI) Master Control Register 0 (BMCR0)	32	RW	0000_0000h
Ch	Bus (AXI) Master Control Register 1 (BMCR1)	32	RW	0000_0000h



- BMCR registers can be used to balance external memory access efficiency, urgency and latency based on requirement of a specific applications. The arbitration logic calculates a SCORE for each request based on the formula and re-ordering the queue.
- BMCR0 controls queue A, BMCR1 controls queue B. For the SDRAM requests, they will be put in queue A at first and executed, when queue A is full, queue B will fetch the SDRAM requests from queue A, then all SDRAM requests will be executed from queue B. So formally, you can think there is one queue (A + B) for all SDRAM requests, the basic execution flow can be guaranteed.
- BMCR0 SCORE formula: SCORE = QOS*WQOS + AGE*WAGE/4 + WSH + WRWS
- BMCR1 SCORE formula: SCORE = QOS*WQOS + AGE*WAGE/4 + WPH + WBR + WRWS

1. QOS stands for AxQOS of AXI bus, and WQOS is the weight factor of QOS.
2. AGE stands for the wait period for each command in queue, and WAGE is the weight factor of AGE.
3. WPH stands for weight of page hit scenario.
4. WBR stands for weight of bank rotation scenario.
5. WRWS stands for weight of slave hit without read/write switch scenario.
6. WSH stands for weight of slave hit without read/write switch scenario.

SEMC re-order



- All requests from different AXI masters will be buffered in SEMC command queue
- SEMC can re-order the requests in the queue for better performance
- BMCR0 and BMCR1 can define the re-order strategy
- SEMC has basic re-order rule to guarantee the access
  - No re-order for same master requests
- DCache requests have different master IDs for read and write
- In default, we enabled the WBR, WRWS, WPH, WAGE and WQOS in BMCRx registers
- WRWS will allow SEMC to re-order the read/write request from different AXI master
- Suppose DCache writes a line to SDRAM, then code need to read the same address, so DCache read the address next to, the write and read requests will be buffered in the SEMC queue and unfortunately SEMC re-orders them and implements the read request in first. The core will fetch wrong data and runs in crash
- More requests in SEMC queue from other master, more delays to DCache requests, more possibilities to re-order the DCache read/write requests
- Modify the BMCR registers to 0x81, so all the requests will be implemented in order strictly, then we fix the crash issue