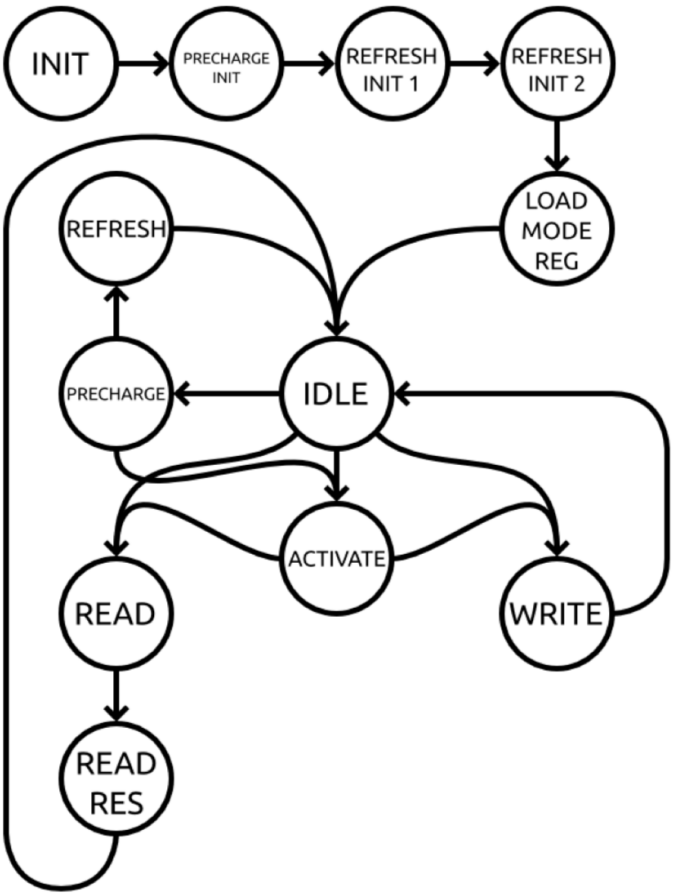
SOC Lab#D - SDRAM report

110061217 王彥智 110590022 陳冠晰 110000107 陳柏翰

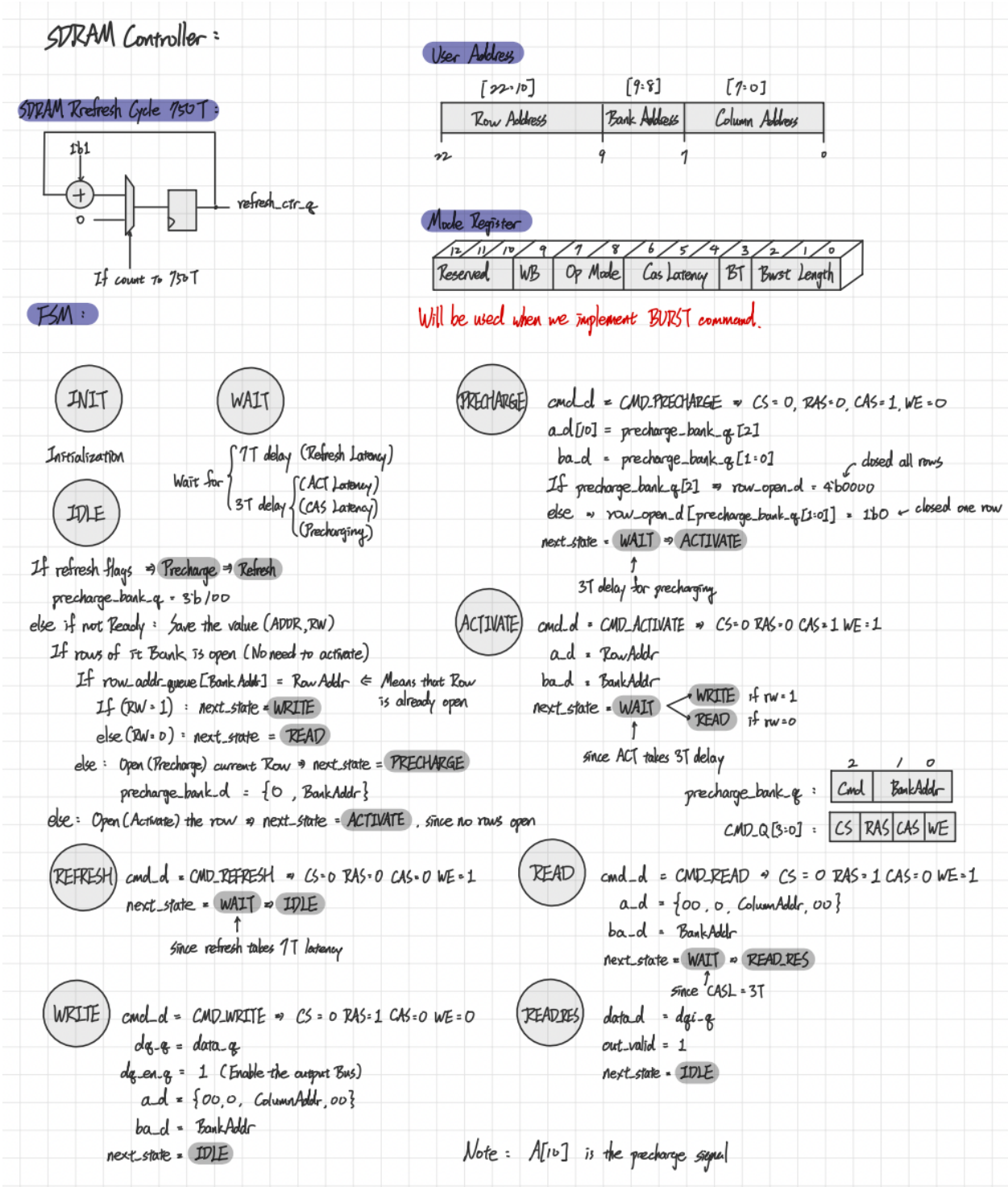
- ## Introduction of SDRAM

  Synchronous Dynamic Random Access Memory (SDRAM) control involves managing data storage and retrieval in synchronized cycles. It utilizes a clock signal for precise timing, enhancing memory access speed. Control includes commands like activate, read, and write, synchronized with the system clock. CAS latency determines the delay between column access and data output. Refresh cycles prevent data loss due to charge leakage. Memory controllers orchestrate these operations, optimizing data flow between the CPU and SDRAM, crucial for high-performance computing. Efficient SDRAM control ensures rapid and synchronized data handling, enhancing overall system responsiveness and computational efficiency.

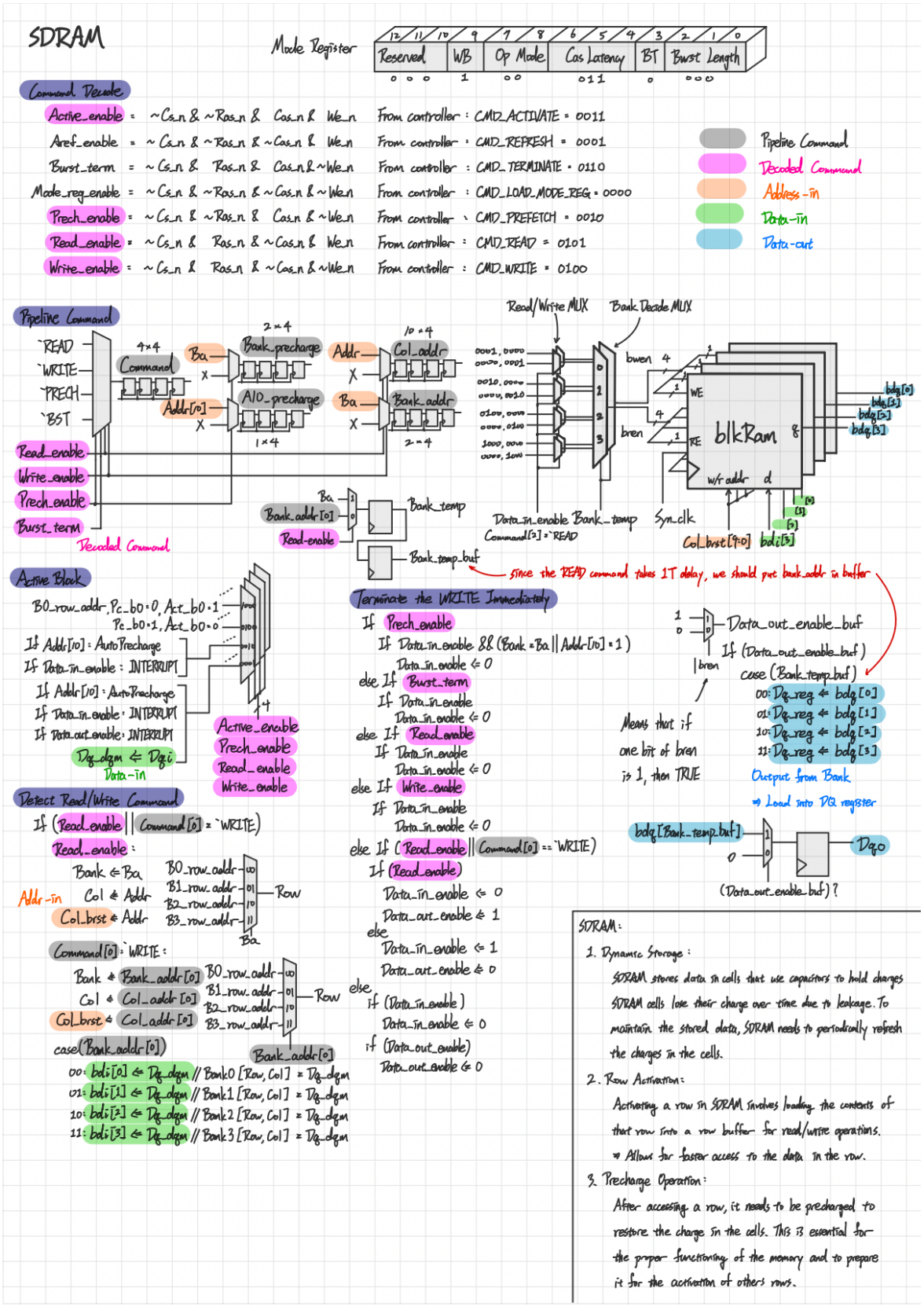


State diagram for SDRAM

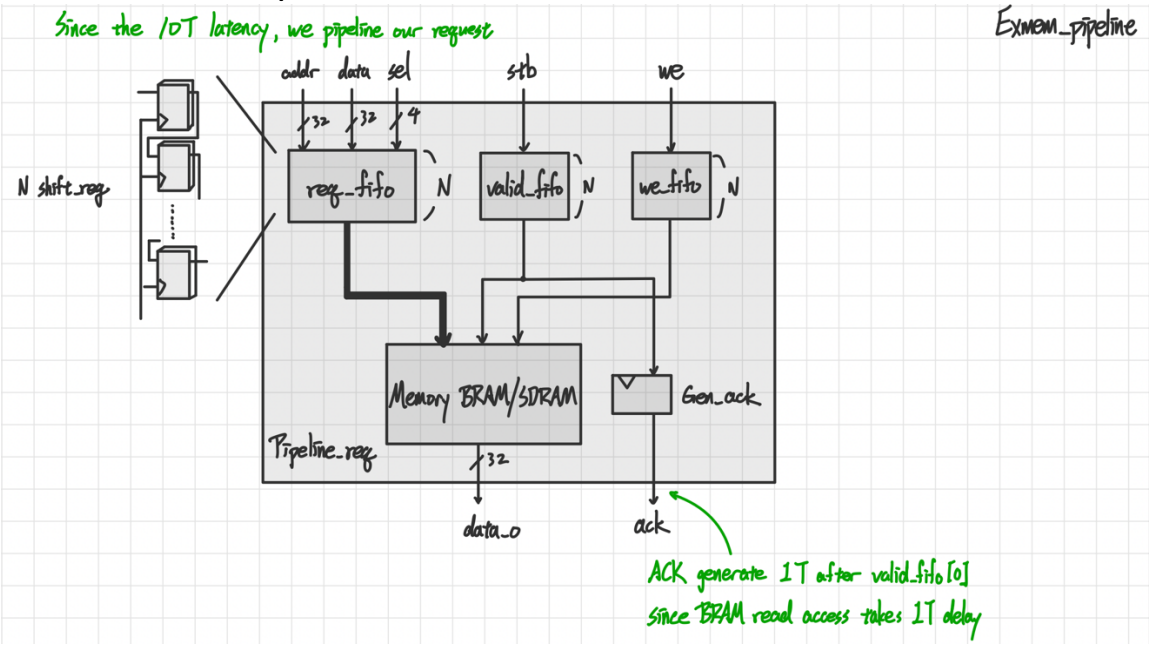
- SDRAM control design



- SDRAM

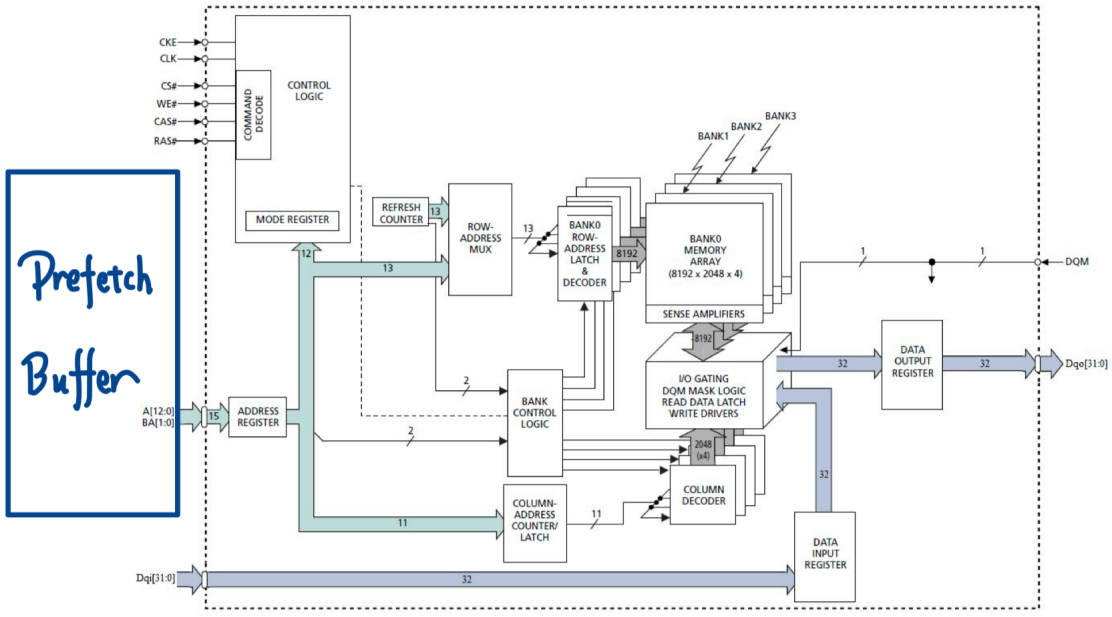

SDRAM
Mode Register
Reserved
WB
Op Mode
Cas Latency
BT
Burst Length
Command Decode
Active_enable = ~Cs_n & ~Ras_n & Cas_n & We_n From controller : CMD_ACTIVATE = 0011
Aref_enable = ~Cs_n & ~Ras_n & ~Cas_n & We_n From controller : CMD_REFRESH = 0001
Burst_term = ~Cs_n & Ras_n & Cas_n & ~We_n From controller : CMD_TERMINATE = 0110
Mode_reg_enable = ~Cs_n & ~Ras_n & ~Cas_n & ~We_n From controller : CMD_LOAD_MODE_REG = 0000
Prech_enable = ~Cs_n & ~Ras_n & Cas_n & ~We_n From controller : CMD_PREFETCH = 0010
Read_enable = ~Cs_n & Ras_n & ~Cas_n & We_n From controller : CMD_READ = 0101
Write_enable = ~Cs_n & Ras_n & ~Cas_n & ~We_n From controller : CMD_WRITE = 0100
Pipeline Command
Decoded Command
Address-in
Data-in
Data-out
Pipeline Command
Read/Write MUX
Bank Decode MUX
blkRam
Decoded Command
Active Block
Terminate the WRITE Immediately
Detect Read/Write Command
Since the READ command takes 1T delay, we should put bank_addr in buffer
Means that if one bit of bren is 1, then TRUE
Output from Bank
⇒ Load into DQ register
SDRAM:
1. Dynamic Storage:
SDRAM stores data in cells that use capacitors to hold charges. SDRAM cells lose their charge over time due to leakage. To maintain the stored data, SDRAM needs to periodically refresh the charges in the cells.
2. Row Activation:
Activating a row in SDRAM involves loading the contents of that row into a row buffer for read/write operations.
⇒ Allows for faster access to the data in the row.
3. Precharge Operation:
After accessing a row, it needs to be precharged to restore the charge in the cells. This is essential for the proper functioning of the memory and to prepare it for the activation of others rows.

- SDRAM FIFO request



Since each read or write has 10T latency, we try to add request FIFO to pipeline the request. The other implementation we can think of is implementing SDRAM with burst.

- Prefetch schematic

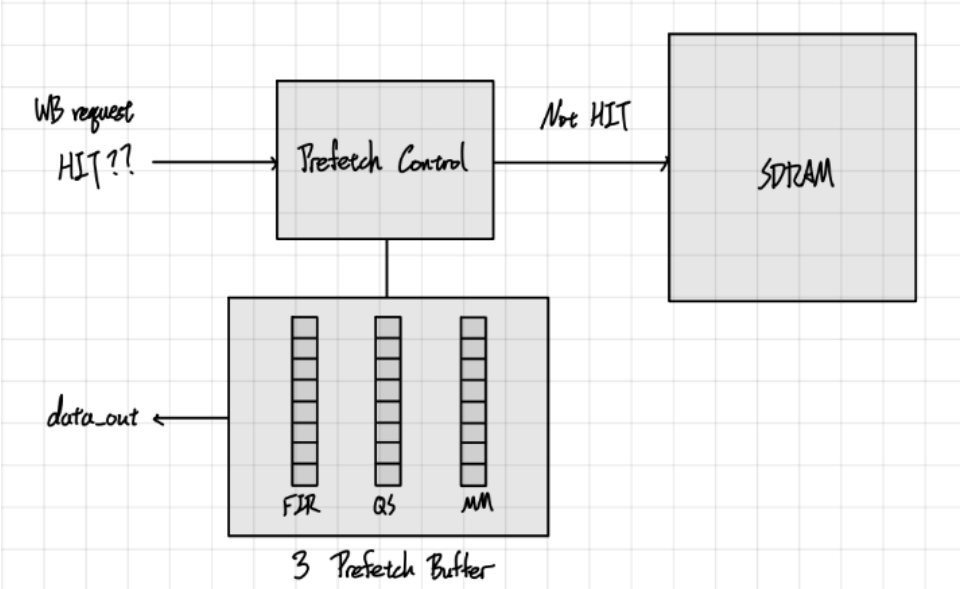


Determine the length of the prefetch buffer. Common prefetch sizes are 4 words, 8 words. Configure the SDRAM controller to use an optimal burst length.
The burst length determines the number of data words that are transferred in a single

burst. A longer burst length can improve memory access performance.
Also, we have to identify the memory access pattern so that we can implement the prefetch buffer with maximum rate of hits, which is the meaning of adding a prefetch buffer.

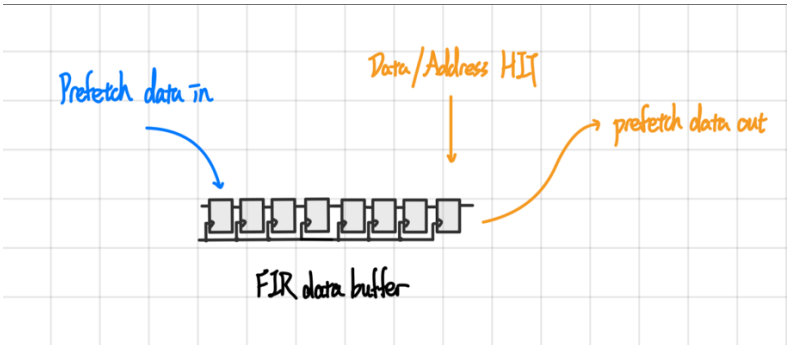
Design consideration

Block diagram

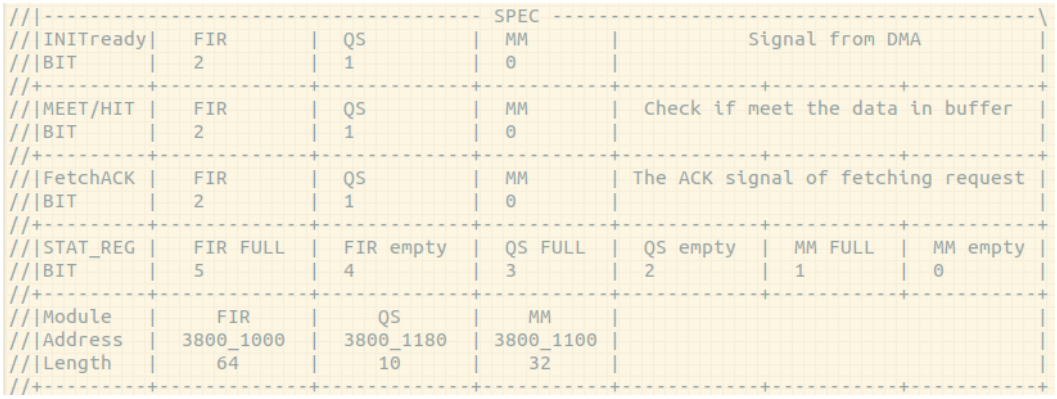


We have 3 prefetch buffers (FIR/MM/QS), here we use FIR buffer to explan our idea. Our prefetch buffer acts like shift registers, it shift out the stored data when the read access came. It prefetch data untill all buffers are filled up before our workload start. If the buffer is empty, controller will send a Empty signal to tell the arbiter to let the priority of that buffer to be the last since the status of that buffer is busy, it is being filled up.

Prefetch buffer



Prefetch controller



SPEC						
INITready BIT	FIR 2	QS 1	MM 0	Signal from DMA		
MEET/HIT BIT	FIR 2	QS 1	MM 0	Check if meet the data in buffer		
FetchACK BIT	FIR 2	QS 1	MM 0	The ACK signal of fetching request		
STAT_REG BIT	FIR FULL 5	FIR empty 4	QS FULL 3	QS empty 2	MM FULL 1	MM empty 0
Module	FIR	QS	MM			
Address	3800_1000	3800_1180	3800_1100			
Length	64	10	32			

Signal Configuration

Above figure shows our design about prefetch controller.

— Setup

- When setup(before all buffers are FULL), data will store in SDRAM first.
- After we got the initial address(sent by DMA) => start prefetch.
- Fill the data until it buffer length, then set the state of that buffer to 'FULL'.
- After all prefetch buffer is 'FULL', call ap_start.

— Running

- If input address meet the saving address => HIT
- If HIT, terminate the wishbone read request by sending the ACK immediately.
- If the buffer is Empty, start prefetch the data from SDRAM into buffer.

- ## Banks interleave for code and data

  We intend to separate the memory for firmware instructions and data for computation. So the bank address should be modified so that the instruction and data are stored in different banks.

- ## Modification the linker

  Address decoder should be modified, since the original decoder use addr[9:8] to decide which bank the data or instructions to put in. However, the instruction of firmware code is larger then the 8 bits address. Therefore we modified the linker of bank from addr[9:8] to addr[13:12] to make more space (8bits to 12bits) to store the assembly instructions.

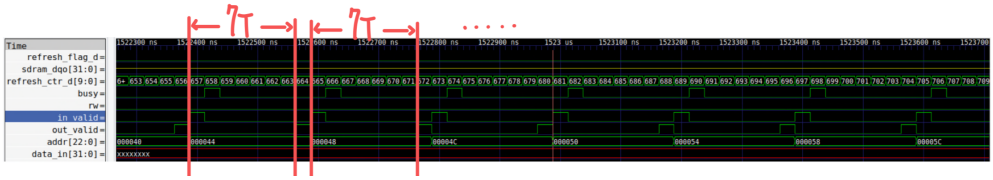
- ## Observe SDRAM access conflicts with SDRAM refresh

  In the SDRAM design, we know that the refresh cycle is 750T, then how does it the access signals and refresh operations collide with each other but still make it work?

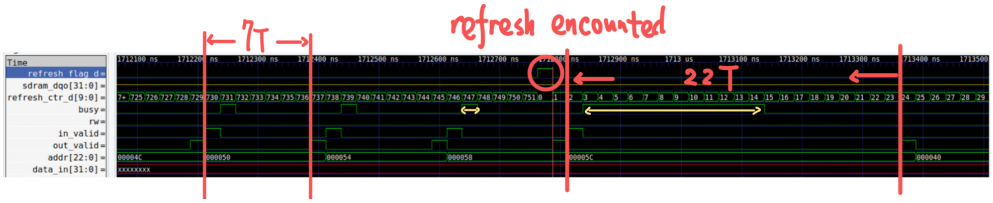
  - Access During Refresh:
    If an SDRAM access (read or write) coincides with a refresh cycle, there may be contention for the memory bus and other resources. This can lead to increased

latency for the access operation. SDRAM controllers are designed to manage these situations by prioritizing access over refresh and scheduling them in a way that minimizes conflicts.

- Refresh Overhead:
  Refresh cycles take away bandwidth and time that could otherwise be used for data access. As such, frequent refresh requirements can impact the effective bandwidth of the SDRAM. The SDRAM controller must strike a balance between ensuring timely refreshes and allowing for efficient data access.
- Waveform

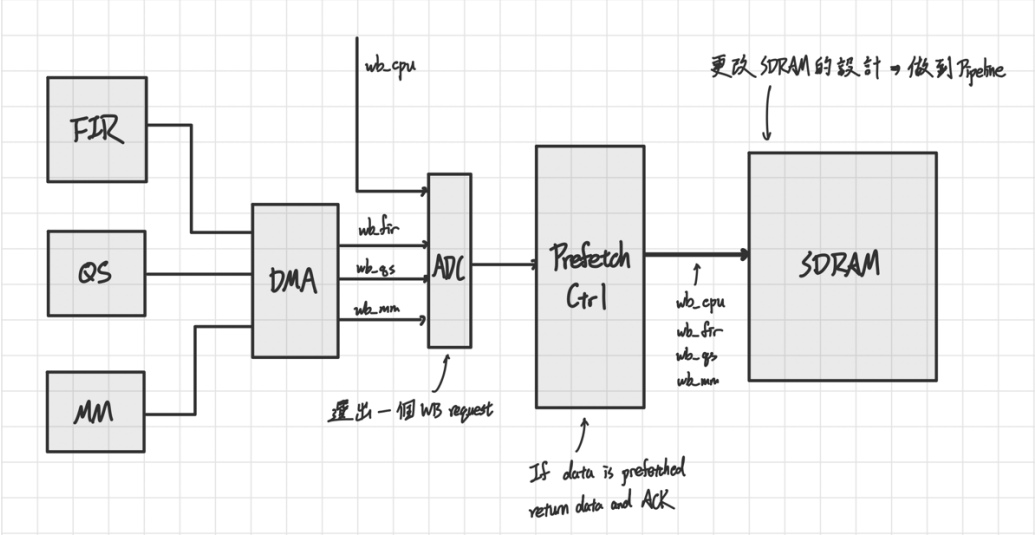


This is the read pattern of SDRAM control when there's no refresh encountered. The latency of read is 7T.



This is the read pattern of SDRAM control when refresh counter= 750. As we can see, the latency of read increases from 7T to 22T since the refresh operation in the SDRAM. The 'busy' signal also lasts longer from 1T to 10T, which is because the refresh time = 10T.

- How do we expect to implement SDRAM in our final project



In the final project, we want to implement the computation (FIR, QS, and MM) using SDRAM to store the instructions and data sets.

Further Optimization

DMA
we plan to design a DMA to improve the execution particularly when it comes to optimizing data transfer between peripherals and memory.

ARB
we also want to design a ARB to distribute the signals from CPU to hardware to make te tasks more efficient.

Prefetch buffer with burst
More, since computation speed heavily dependent on the data transfer time, this is the place we are most interested in optimization. Prefetch buffer with high accurate of hit pattern can greatly reduce the time for SDRAM latency to make the task faster.

- Summary

Performance

Computation – FIR, QS, MM

	Software	Hardware without prefetch	Hardware with prefetch
MM	55303	756	X
FIR	65890	1471	893
QS	14394	315	X
total	135587	1570	X

73.2X
Faster than software MM

46.5X
Faster than software FIR

45.7X
Faster than software QS

Overall performance Optimization

86.4X
Faster than original design