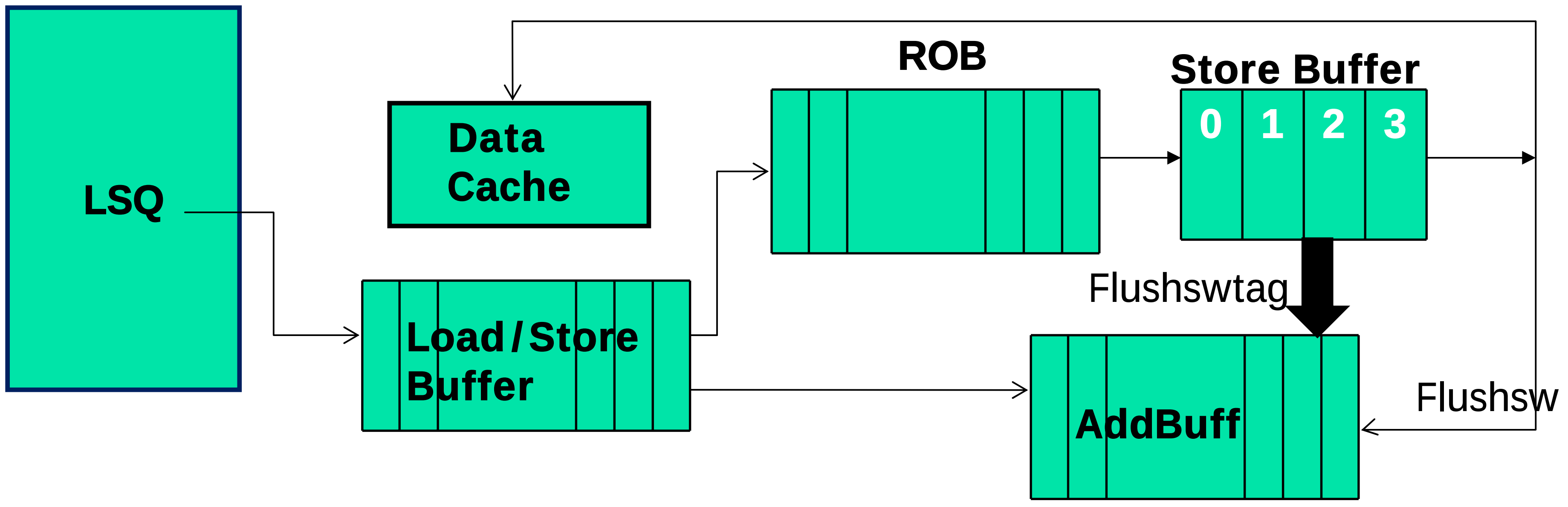
STORE BUFFER

Overview

- What the purpose of having a store buffer (SB)?
  - When a store instruction reaches the top of the ROB and is ready to writes its data to cache, the cache could be busy serving pervious store instructions. Instead of holding other completed instructions in the ROB while waiting for the cache to become available (inefficient design), we put the store instructions in a separate buffer and allow other instructions to commit. This separate buffer is called store buffer (SB).
- Store buffer has four entries indexed as (0,1,2,3). Each entry is sub-divided into four fields as shown below.
- Store buffer entry(3) holds the oldest store instruction that will write to the cache as the cache becomes available. That is why SB is filled up from entry 3 down to entry 0.

Valid bit(1 bit)	Data(32 bit)	Address(32 bit)	SB Tag(2bit)

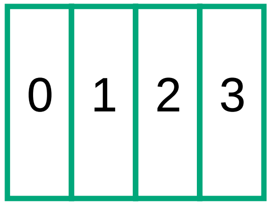
Block Diagram



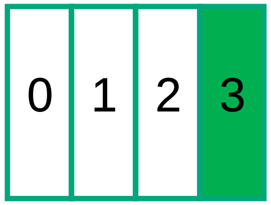
Operation

- SB Allocation:
  - When a completed "SW" is at the top of ROB, we need to save the address and data of the "SW" until it writes to the cache. If the SB is full then we need to stall the commit and wait until the oldest "SW" completes writing to the cache and releases its SB entry.
  - A counter is used to point to the next free entry in SB. The counter has the following properties:
    - 2-bit up/down saturating counter initialized to "11" at Reset which is the first SB entry that we fill.
    - Each time a store instruction joins the SB, the counter is decremented by 1 to point to the next free entry except when the counter value is 0 (Saturating Counter). In addition, we need to make sure that the "SW" in entry 3 did not complete yet, because if it is completed the current entries will shift one location upward and the current counter value will point to the next free entry without decrementing.
    - Each time a store instruction leaves the SB (finish writing to the cache) , the counter is incremented by 1 except when the counter value is 3 (Saturating Counter) because all entries will shift one location upward. In addition, we need to check that no "SW" instruction is added to the SB at the same time, because in that case no need to increment the counter as it will be already pointing to the next free entry.

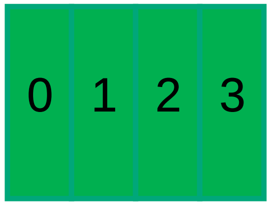
Example



Counter = 3 (SB is Empty)



Counter = 2



Counter = 0 (SB is Full)

Operation

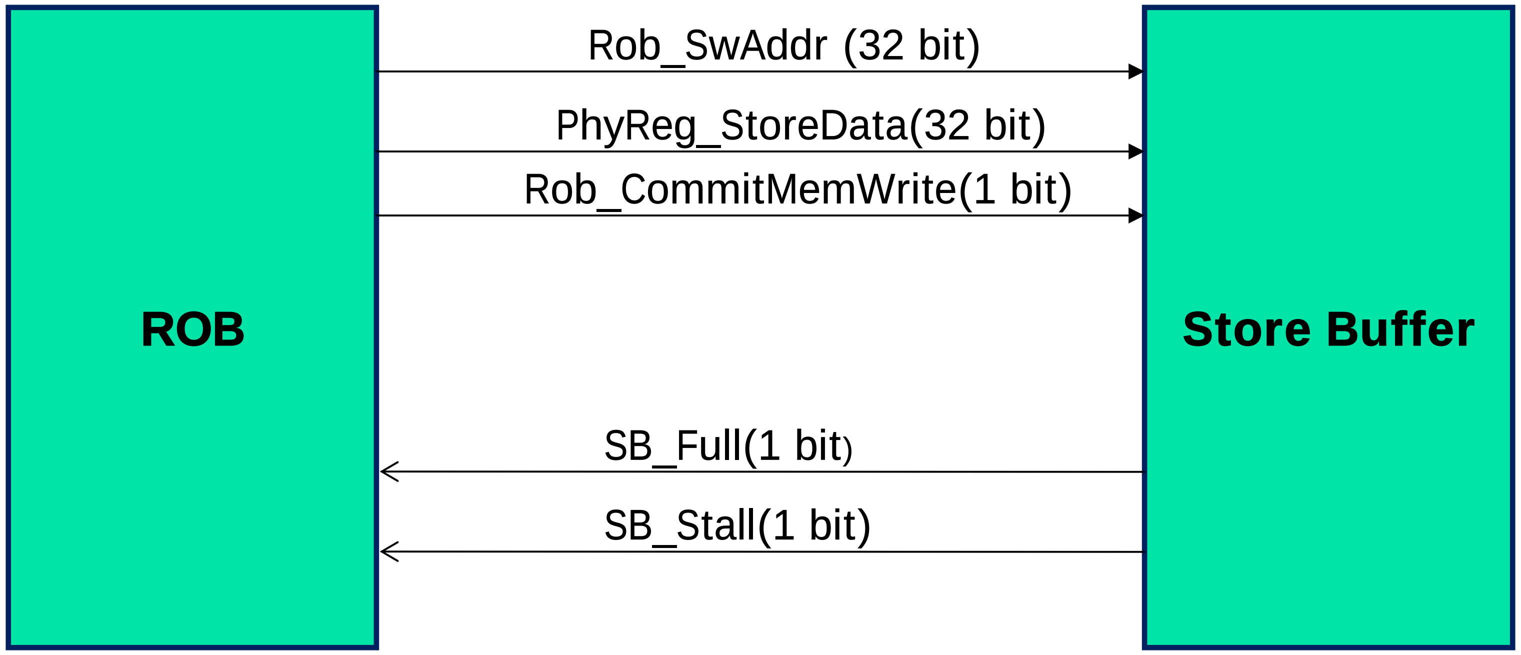
- Fill the allocated SB entry:
  - “SW” Data (32-bit): we read the contents of the physical register mapped to $Rt from the physical register file and save it in the data field of the assigned SB entry (This indicates that the data is not carried with the instruction in the ROB which reduces ROB entry width).
  - “SW” Address (32-bit): each ROB entry has a 32 bit field to hold the address for “SW” instructions, we simply read the address from the ROB and save it in the address field of the assigned SB entry.
  - Valid (1-bit): we need to set the valid bit to “1” to indicate that the entry has a valid store instruction.
  - SBTag (2-bit): This tag is used to flush the associated entry in address buffer when “SW” completes writing its data to the cache. The tag is generated using dedicated 2-bit counter (SBTag_counter) initialized at “00”. Each time we assign a new SB entry, we save the current counter value in the Tag field of that entry and then we increment the SBTag_counter.

Operation

- Shifting SB entries:
  - At the positive edge of the clock, if the data cache is free (Done writing the data of entry(3) to the cache), we shift SB entries (0,1,2) one entry upward provided that they contain valid store instructions.
- Data Write Completion:
  - As the data cache takes time for data to be written, we present the SB_Flushsw signal when memory write is completed which is when WriteDone signal of data cache becomes high. SB_Flushsw signal is used to enable the flush of the corresponding entry in address buffer (The entry that holds the same SBTag).
  - The SBFlushswTag is the SBTag of the “SW” that is being written into the data cache. It is kept in a register (DCETag) and presented later when the SB_Flushsw becomes high.

Interface with other units..

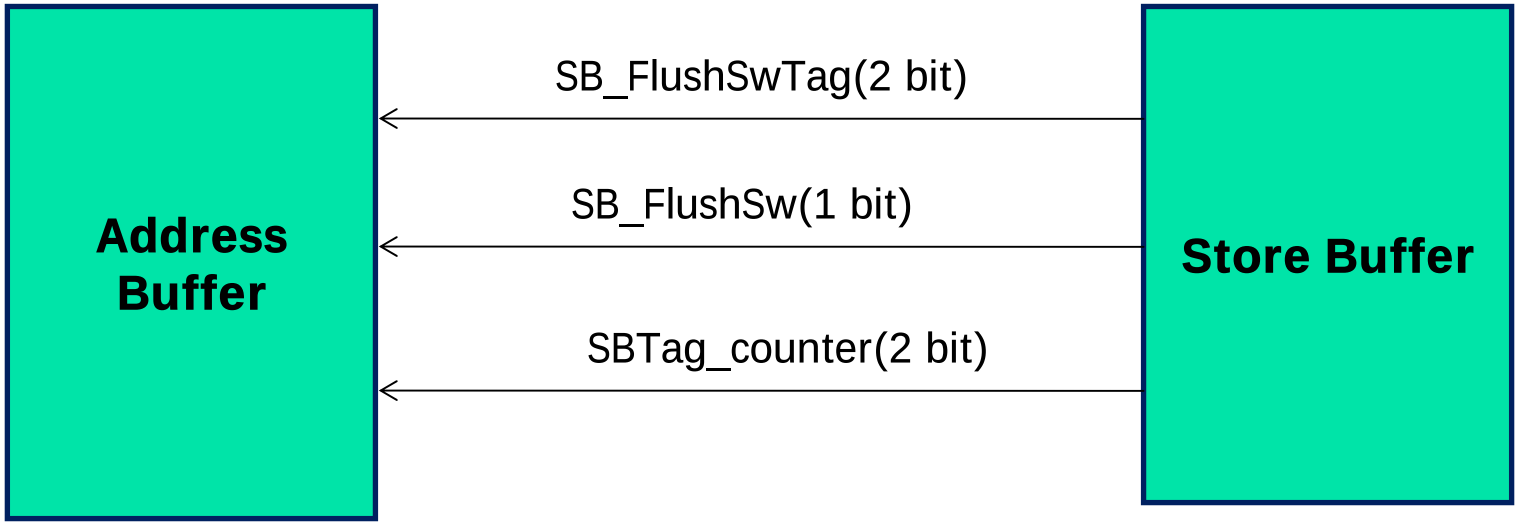
Interaction with ROB



Interaction with ROB

- RobSwAddr (32-bit): Represents the address of the store instruction at the top of the ROB.
- PhyReg_StoreData (32-bit): Represents the store data (value of register $rt). The data is actually read from the physical register file and not from ROB.
- Rob_CommitMemWrite(1 bit): Indicates that the instruction at the top of ROB is a “SW” instruction and it is completed.
- SB_Full(1 bit) and SB_Stall (1 bit): These two are always set together, as whenever the SB is full no more entries can be added and hence SB_Stall will be active too. Basically, we only need one of them.

Interaction with Address Buffer



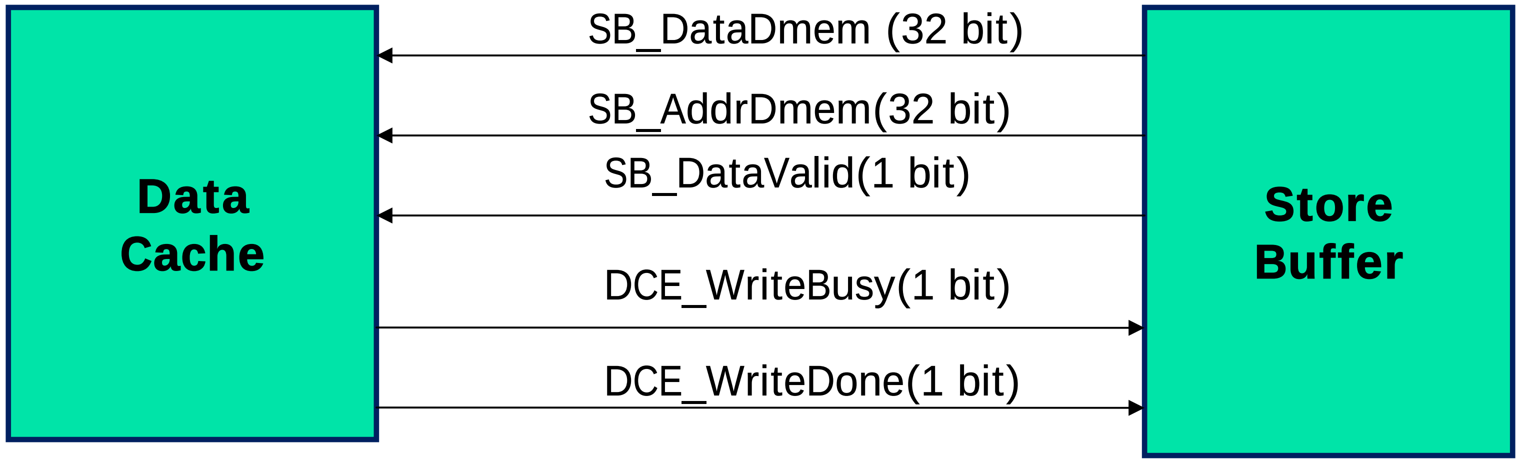
Interaction with Address Buffer

- SB_FlushSwTag: This is the SB tag of the “SW” which is currently being written into data cache and which needs to be flushed from the address buffer.

- SB_FlushSw: This signal enables the flushing of the content of the corresponding address buffer entry.

- SBTag_counter: This signal represents the SB Tag of the “SW” joining the SB. This tag must also be stored in the corresponding entry in the address buffer in order to be used when we flush this entry as the “SW” finishes writing to the data cache.

With Data Cache



- SB_DataDmem: The data contents of entry(3) in the store buffer.
- SB_AddrDmem: The Address contents of entry(3) in the store buffer
- SB_DataValid: This signal indicates that the data in entry(3) is valid to be written to the data cache.
- DCE_WriteBusy, DCE_WriteDone are used to acknowledge the memory write completion.