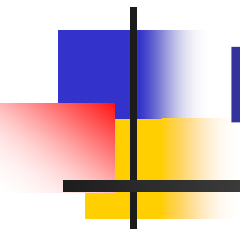


LOAD STORE QUEUE

Load store queue

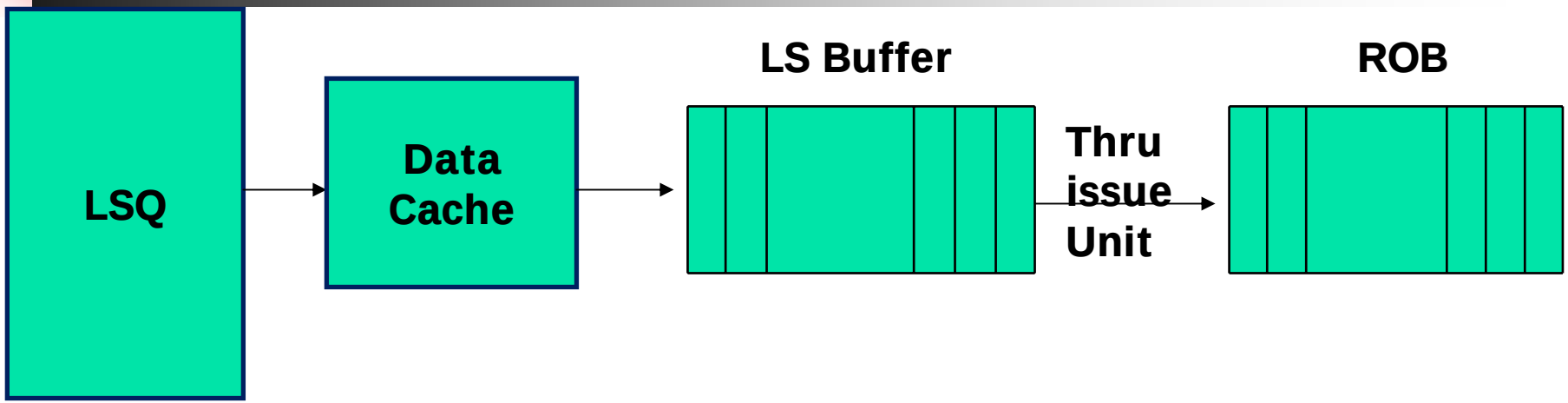
- It interfaces with Dispatch, Reorder Buffer, Store buffer, Physical Register File, CDB and Issue unit as well as functional units.
- Load/store queue has Eight entries and each storing a valid bit, Rs tag, Rd tag, ROB tag, load/store Addresses, Opcode, instruction ready bit and also generates an Address ready bit signal.

InstructValidbit(1 bit)	Rsdatavalid(1bit)	Opcode(1bit)	Addressreadybit(1 bit)	ROBtag(5 bit)	Rstag(6 bit)	Rdtag(6bit)	Addressreg(32 bit)

Load Store queue

- In this design, we do not carry the data through the queues but we carry the tags of corresponding registers and their ready bits. Once the address is calculated, we issue the instruction. The register tag is given to the Physical register file and we asynchronously read the data from the PRF and then we calculate the address for the Load /Store words.
- We do forwarding from the CDB and not from the ouput of the integer queue or other queues. This is equivalent to watching to the write port of the physical register file. There by we are wasting one clocking in forwarding over the CDB (*this is one major difference between load store queue and other issue queues*).

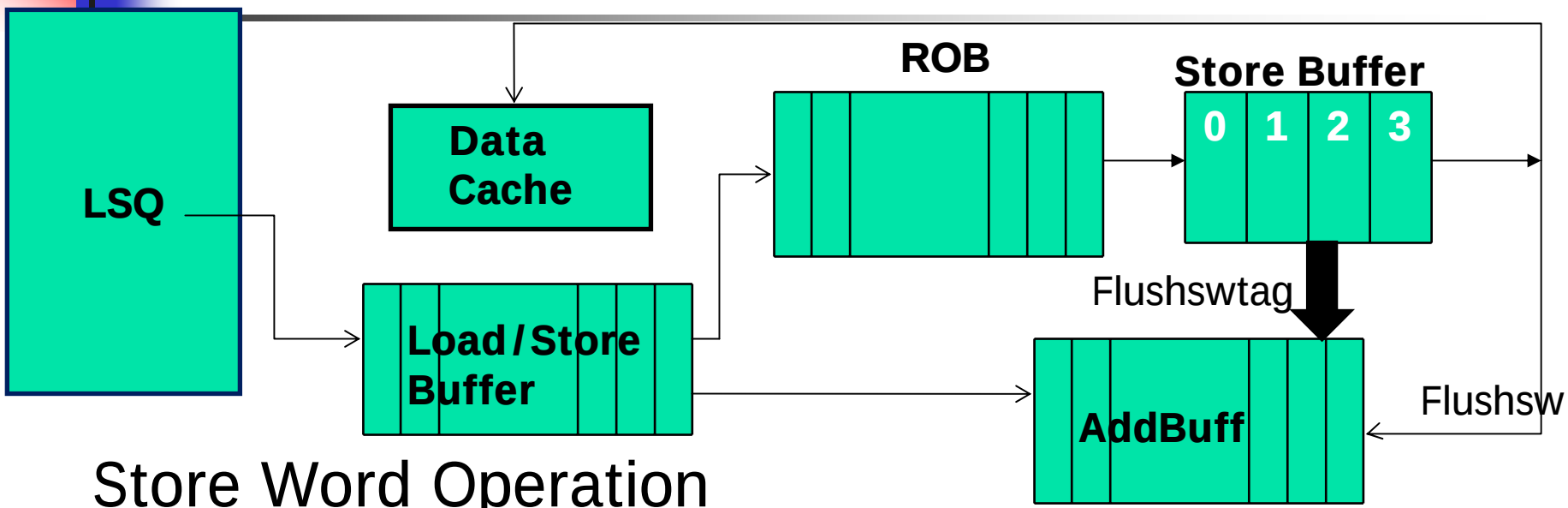
Block Diagram and operation



Load word operation:

- Load word accesses the data cache memory and puts the data fetched into the Load/Store buffer which in turn put it on the CDB (governed by the issue unit) and it is then passed on to the RoB where it is later written into the corresponding Physical registers on committing.

Block Diagram and operation



Store Word Operation

- Once the address is ready and after satisfying all the memory disambiguation in the queue ,store word is issued from the queue. While leaving the queue the Address Buffer plays its part of dealing with the memory disambiguation (RAW hazard) aspect.
- The issued store word is put into Load/Store buffer and then on the CDB (governed by issue unit), then it reaches the ROB.

Block Diagram and operation

- The issued store word is carried through the ROB and when the store word reaches the top of the ROB and then it goes into store buffer
- Then it is written into data cache based on latency cycles given by the cache emulator.
- Once the data is written into the data cache, the store word at the top of the store buffer is then written into the data cache.
- Whenever there is a store word issued from the queue, the address buffer is updated (refer to address buffer slides for more details).
- Once store word is committed from the top of the store buffer, the corresponding entry in the address buffer is invalidated.

Memory Disambiguation

- **Write After Read (WAR)**

This is taken care by the ROB Buffer . Since Store Word writes once it comes to the top of ROB ,So Load word before it would have completed . So no need to have any condition about it however we have a extra condition that does not let the Sw skips Lw until Lw address is ready so that to keep count of how many “sw” having same address as “lw” have skipped it.

2	Sw	2000
1	Lw	2000

- **Write After Write (WAW)**

As explained in WAR, similarly WAW is handled by the in order completion in ROB

2	Sw	2000
1	Sw	2000

Memory Disambiguation

Read after Write (RAW)

- This is done by not letting “lw” go ahead of “sw” having same address. Now Since “sw” can skip “lw” having same address at a given time we can have more than one entries in Addrbuff against “sw” having same destination address .
- For an instruction being a valid "lw" it can only be issued when all the "sw"(with same address) in front of it had comitted. This case is substantiated by address match number being less than issuecounter signal which indicates that all the "sw" that were issued earlier have comitted so one can issue the “lw”.
- For an instruction being a valid "sw" it can be issued only if all the "lw" in front of it have their *address ready*, this precaution is needed because you need to store the address of any bypassing sw (for any lw) if the address matches

Memory Disambiguation

- Now Address Buff will have two entries after both store have been issued by queues . Since one store word has skipped the load word the Addrcounter for Lw = 1. Now when Load word is ready . It will check for number of matched in Address Buff .No of matches = 2 .Since this is greater than AddrCounter this load word will not get issued by Queue.
- After Some time 1st store word gets committed from ROB and subsequently completed from store buffer. Now the load word in queue will Have number of Address Macth = 1 which is equal to or less than Its Address Counter . So now it will get issued. This is how AddrBuff is used for RAW hazard .

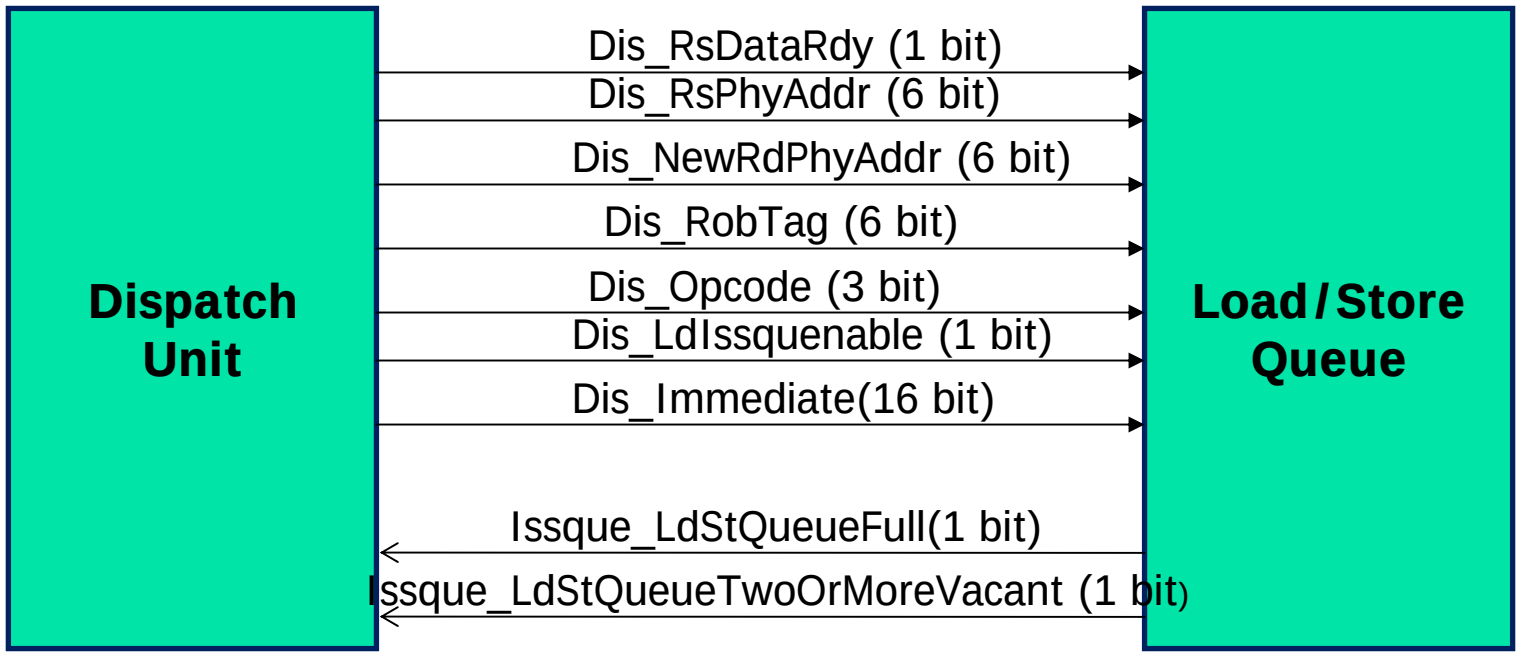
3	Sw	9000
2	Lw	9000
1	Sw	9000

3	9000
1	9000

3	9000

Interface with other units..

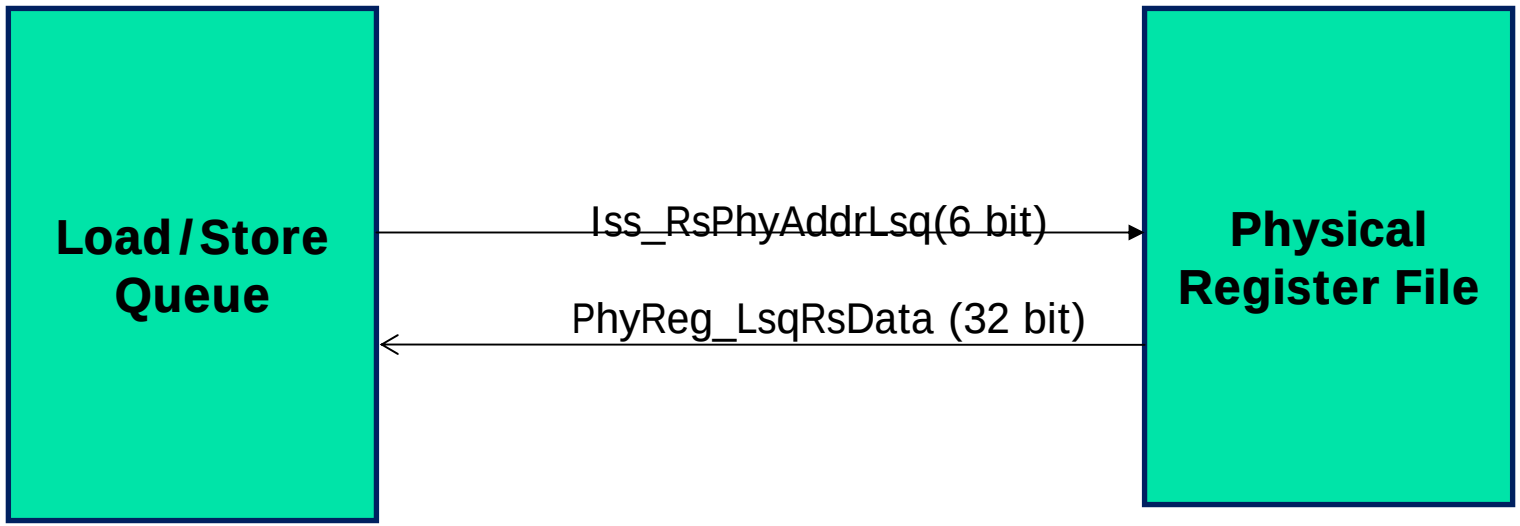
With Dispatch



With Dispatch

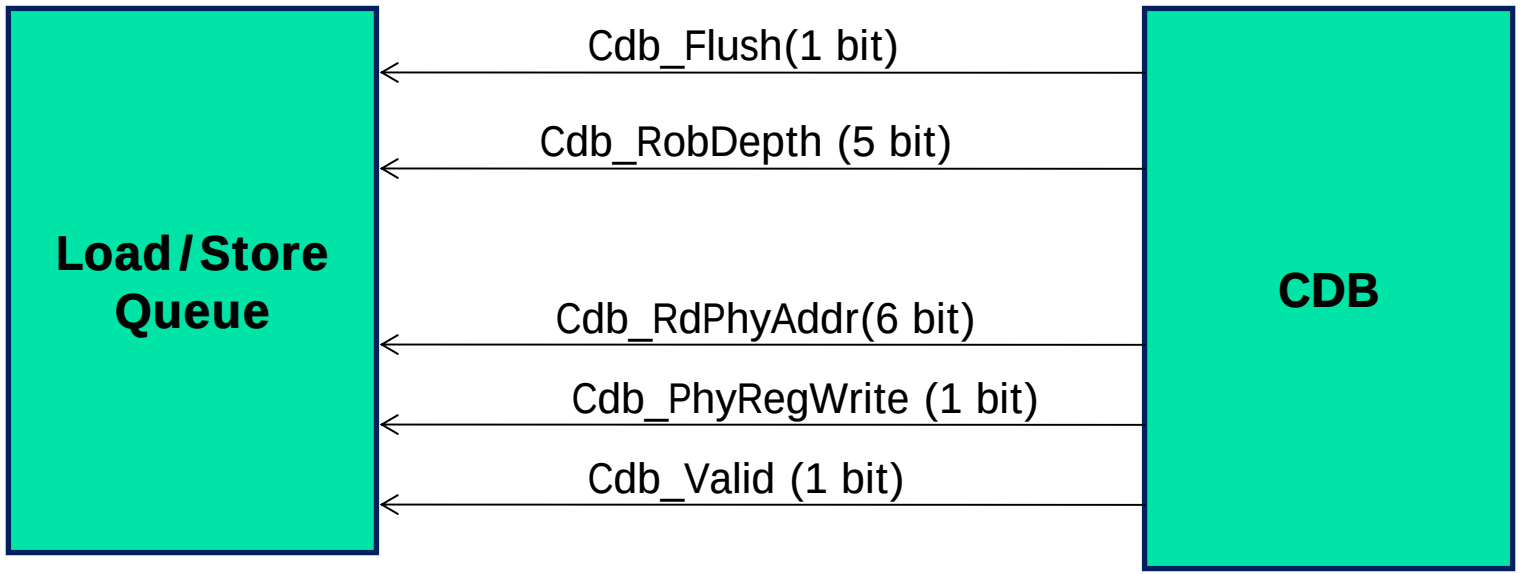
- Dis_RsDataRdy and Dis_RsPhyAddr gives ready bit and physical register address of Rs for the instruction
- Dis_NewRdPhyAddr gives the new Rd(destination) physical address for the instruction and stored in the queue.
- Dis_RobTag, Dis_opcode gives the rob tag and 1 bit opcode which tells whether it is store or load word. Opcode=1 implies the instruction is a store word.
- Dis_LdIssquenable is enabled whenever the queue is not full and a new entry is to be written into the queue
- Dis_Immediate is the 16 bit immediate address of the Load/Store words.
- Issque_LdStQueueFull tells us whether the queue is full or not full.
- Issque_LdStQueueTwoOrMoreVacant is a signal which tels us that the queue has two or more vacant slots left in the queue.

With PRF



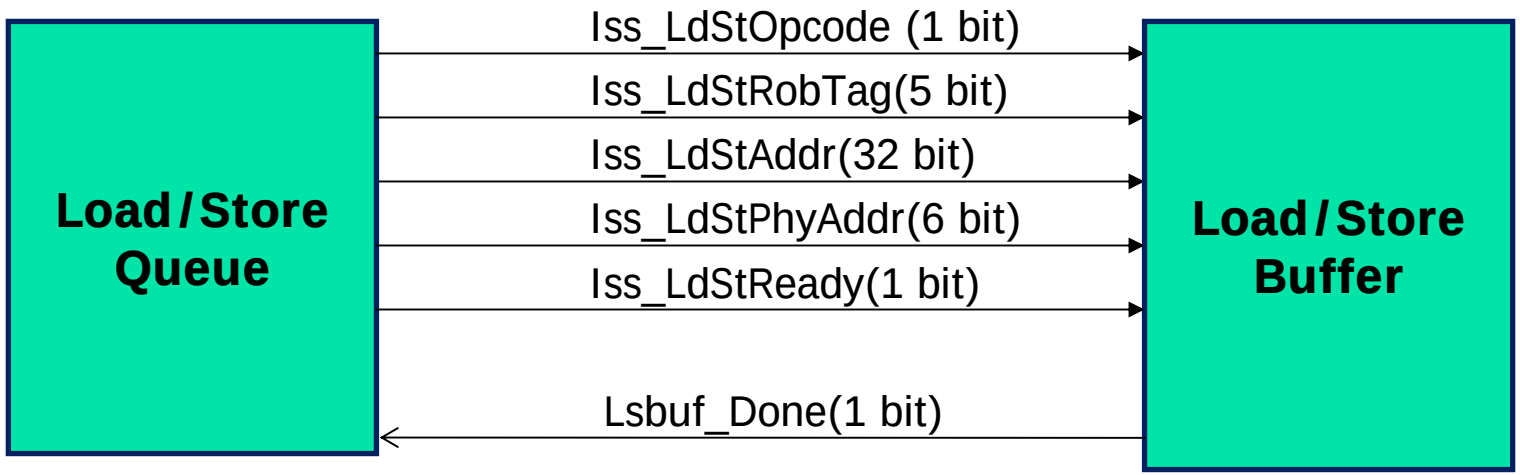
- Iss_RsPhyAddrsq: The Rs tag is given to the PRF on this signal one at a time to calculate the address.
- PhyReg_LsqRsData: The Data for the corresponding Rs register is presented asynchronously in the same cycle.

With CDB



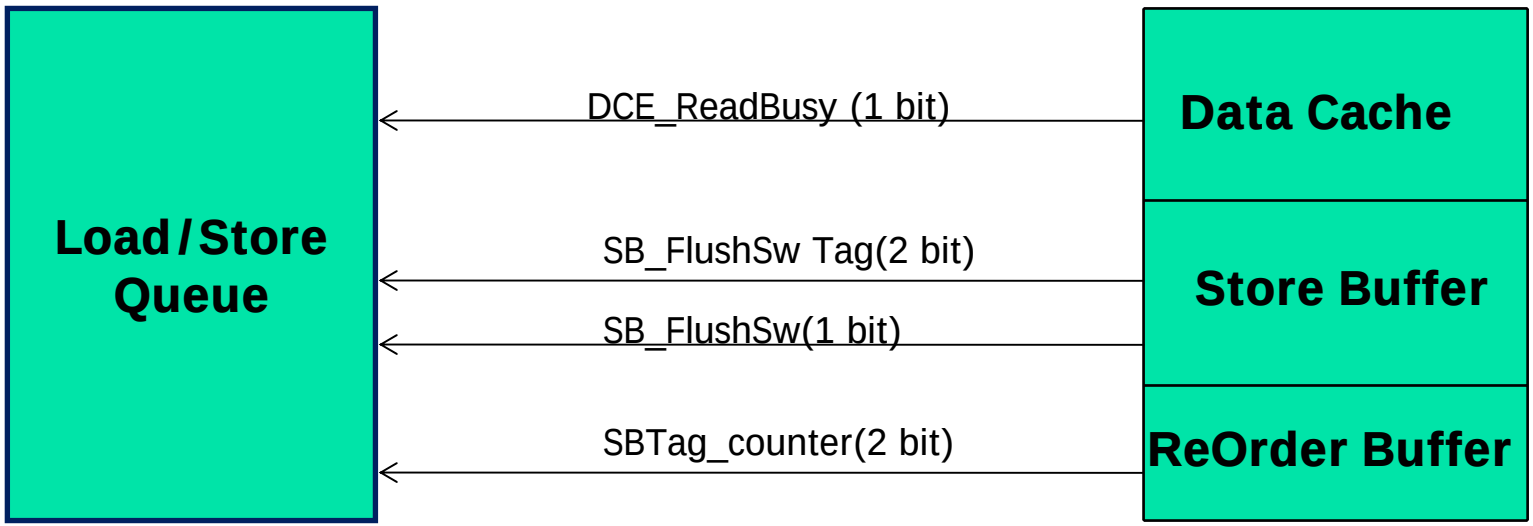
- All the above signals from the Cdb is used in the LSQ cntrl to match the Rd address(Cdb_RdPhyAddr) and the valid(Cdb_Valid) signal and a regwrite signal(Cdb_PhyRegWrite) to update the redy bits of the Queue entries.
- Cdb_Flush and Cdb_RobDepth is used when there is branch misprediction.

With Load/Store Buffer



- All the above signals Iss_LdStOpcode tells whether it's a Load or store word, Iss_LdStRobTag tells the rob tag of instruction, Iss_LdStAddr gives the address calculated, Iss_LdStPhyAddr tells the physical address where data has to be written or read . All the signals are given to the buffer in case of store word where as for load word it is given to Data cache and then the ouput of the cache is put into the Load store buffer.
- Lsbuf_Done=1 tells that Load/Store buffer is not full.

With Other Units



- All the above signals SB_FlushSw and SB_FlushSwTag tells which entry in the address buffer is to be invalidated when a store word which commits.
- SBTag_counter is used for flushing the address buffer when the “SW” finish writing to the data cache..
- DCE_ReadBusy tells that data cache is reading and is currently busy.

(refer to Address Buffer and Store buffer slides)

Load/store Control module signals

- Issuequecounter: This counter is used keep a count of “sw”s that skipped a “lw” having the same address for each of 8 entries in the Load/Store queue.
- BufferRstag and BufferRdtag : These are the tags of the Rs and Rd which is carried in the queue.Opcode is the 1 bit signal which is used to distinguish between “lw” and “sw”s.If opcode =1 then it’s a store word else it’s a load word.
- AddrReadybit tells whether the entries in the queue has its address ready or not.
- Rsdatavalidbit and InstrValidbit gives the validity of the Rs data and instruction in the queue.
- Iss_LdStIssued is “1” when an instruction Load/Store queue is issued. Dis_LdIssquenable is “1” when there is a instruction presented to the queue from the dispatch unit.
- Cdb_Flush, Rob_Topptr, Cdb_Robdepth are used to invalidate entries in the queue when there is a branch misprediction.
- Cdb_RdPhyAddr, Cdb_valid, Cdb_PhyRegWrite :These signals are used for data forwarding and updating the Rsreadybit in queue.We compare the Rstags in the queue with that of the Cdb_RdPhyAddr and Cdb_PhyRegWrite=‘1’ meaning a regwrite instruction and a valid one (Cdb_valid=‘1’) ,if there is a match then we update the corresponding Rs bit in the queue.

Load/store Control module signals

- Outselect indcates which entry in the 8–entry queue is to be issued.
- Sel1Rs is the control signal used for updating the Rsreadybits in the queue by data forwarding from the CDB.
- Flush bits are the control signals given to the address buffer to remove its entries when there is misprediction.
- Sel0 is a control signal used when there is an new entry is being pushed into top of the queue ,En is used to move the queue.
- AddrUpdate, AddrUpdatesel are the control signals which are used to update the address of the entry in the queue. AddrUpdate will consider its own address field for the address calculation while AddrUpdatesel will use the prior entry to calculate the address.
- Addrbuffful is used to indicate that the address buffer is full. AddrmatchNum gives no of address matches for each entries of the queue with address buffer entries.
- SB_FlushSw, SB_FlushSwTag are used flush the contents of address buffer once the “sw” commits from the top of the buffer and we wait until we completely finish writing into the data cache buffer.