PHYSICAL REGISTER FILE

Read/Write

32-bit wide 48 deep register file. $0 is never written during circuit operation. So it is replaced with hard wired 0s by synthesis tool

It has 8 read port and 1 write port. Reading is asynchronous and writing is synchronous.

These read ports are needed by various execution units - 2 each for Integer, Multiply, and divide, 1 for LSQ for load/store address computation and 1 for store buffer for store data

Writing is done by the CDB whenever a register-writing instruction appears on it

This regfile is internally forwarded; so if the required data appears on the CDB it is forwarded to the read port(s) in the same clock

Data is read by an execution unit from the register file in the same clock in which the instruction is issued to it – hence the Rs/Rt register information comes from the issue units but the data goes to the execution units directly.

Ready Bit

Each physical register has a corresponding ready bit associated with it. Thus there is an array of 48 ready bits total

When an instruction is dispatched the ready bit for new physical register corresponding to the Rd register is made 0 (i.e. its value is not ready).

When an instruction is on CDB the ready bit for new physical register corresponding to the Rd register is made 1 (i.e. now the value of this physical register file is ready and any instructions depending on this register may now be issued)

Hence there are two write ports for the ready array (one for dispatch, one for commit), and two read ports (one for Rs and one for Rt)

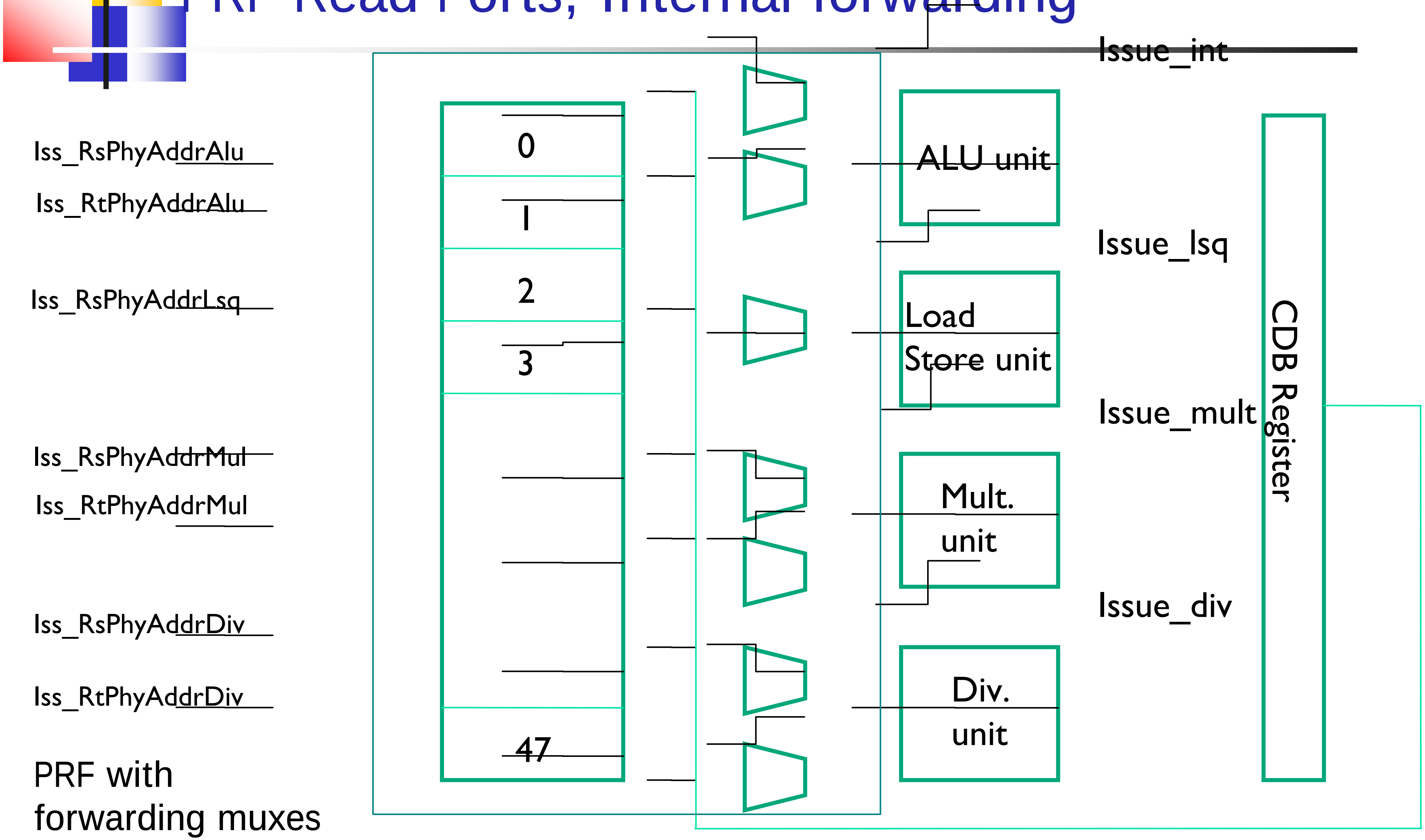
This ready bit is used by issue queues to decide whether or not to declare an instruction ready for issue.

Operations in Physical Reg. File

- Flushing or stalling does not affect the Physical Register File.
- The Frl takes care of Flushing operation*.
- Physical Reg. Ready signal only read by Dispatch and all other units look at the write port for the RtReady and RsReady signal.
- Physical Reg. Ready signal invalidated for corresponding New Rd Physical Register issued by Dispatch.

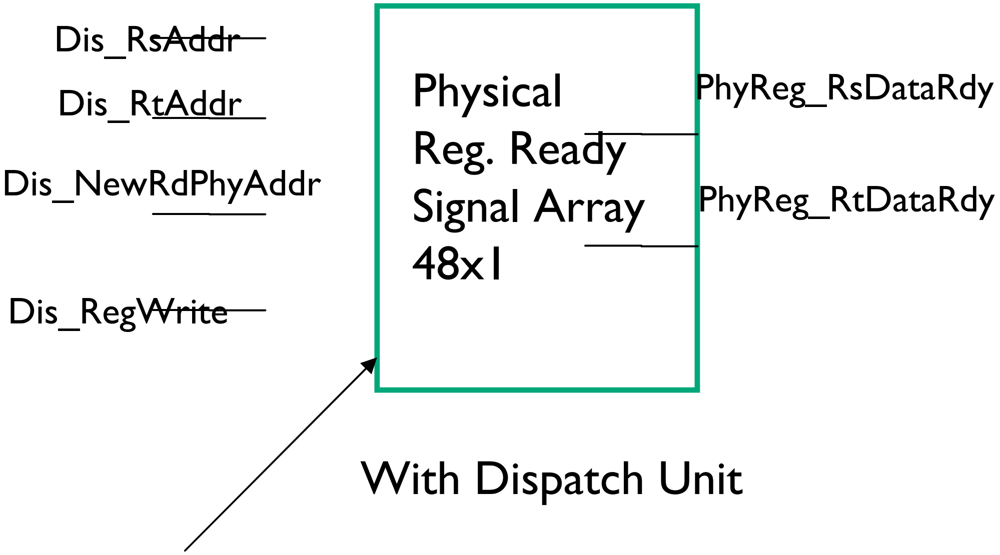
NOTE: * Whenever a new physical register is assigned for a "rd" register of an instruction , the physical register file needs to set the "ready" bit "0" corresponding to that physical register. It is because on flushing we are not invalidating the ready bits of the physical registers which get freed (as it will call for invalidating multiple locations simultaneously) Thus **Dis_NewRdPhyAddr** tells the physical register to be assigned. **Dis_RegWrite** is needed to validate the operation as address pins may contain the address of used physical register when FRL is empty.

PRF Read Ports, Internal forwarding



PRF with forwarding muxes

Signals interfacing with Prf Unit.



Dis_RsAddr

Dis_RtAddr

Dis_NewRdPhyAddr

Dis_RegWrite

Physical Reg. Ready Signal Array 48x1

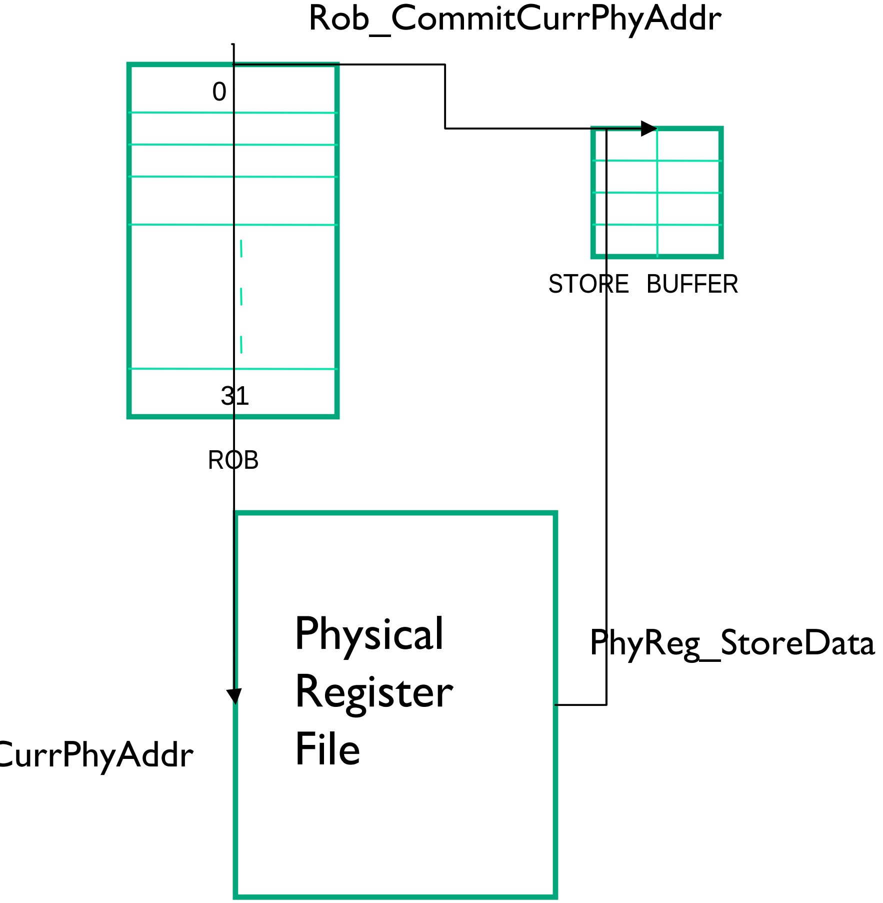
PhyReg_RsDataRdy

PhyReg_RtDataRdy

With Dispatch Unit

2 read ports and 1 write port for dispatch

1 write port for CDB



Rob_CommitCurrPhyAddr

0

31

ROB

STORE BUFFER

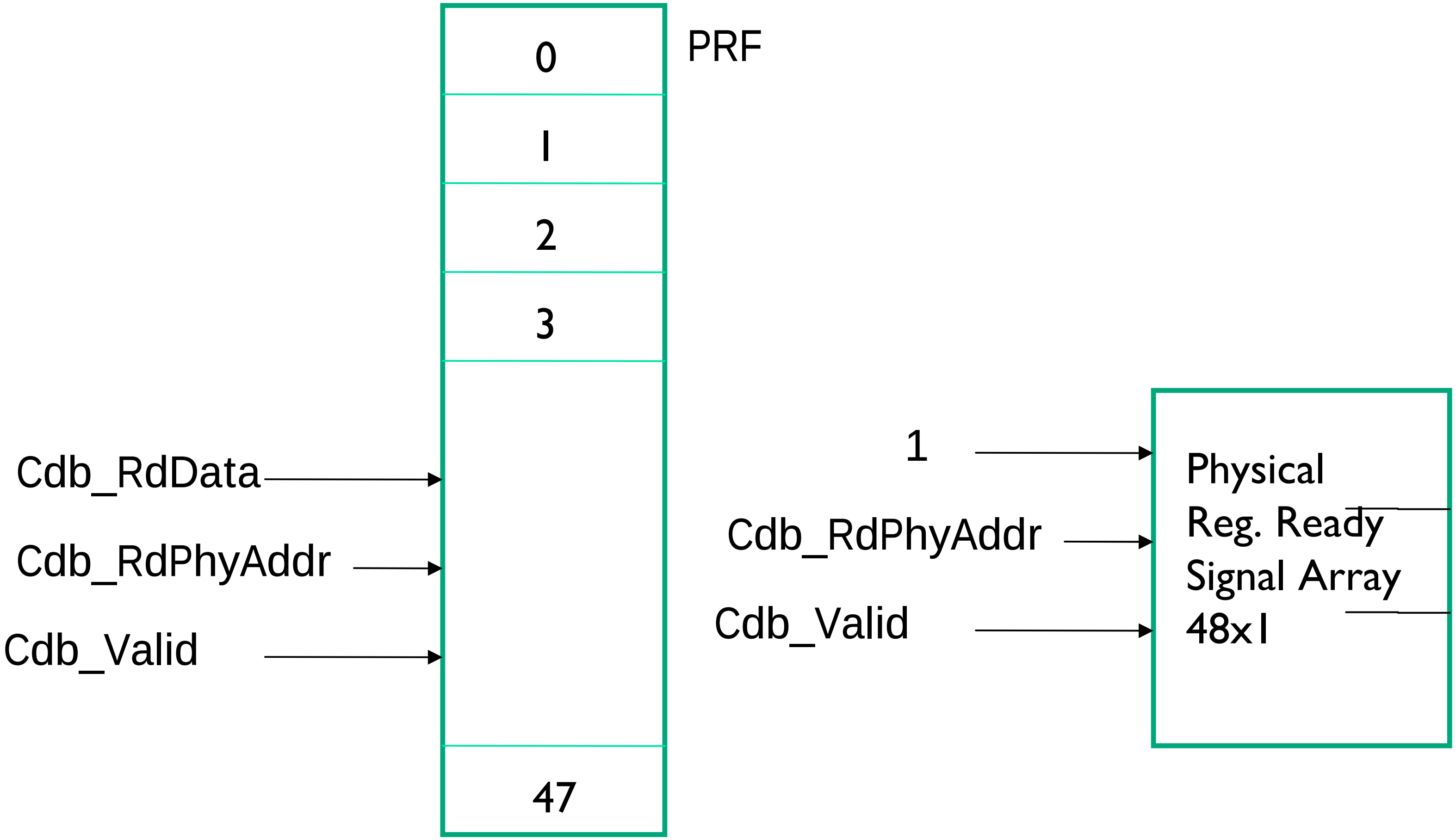
Physical Register File

PhyReg_StoreData

Rob_CommitCurrPhyAddr

With Store Buffer

CDB writes to PRF and the ready signal array



PRF

0

1

2

3

47

Cdb_RdData

Cdb_RdPhyAddr

Cdb_Valid

1

Cdb_RdPhyAddr

Cdb_Valid

Physical Reg. Ready Signal Array 48x1