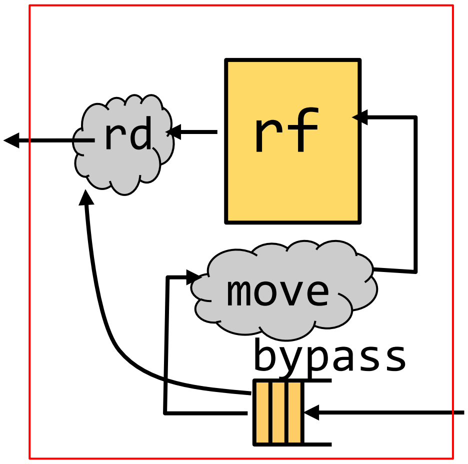
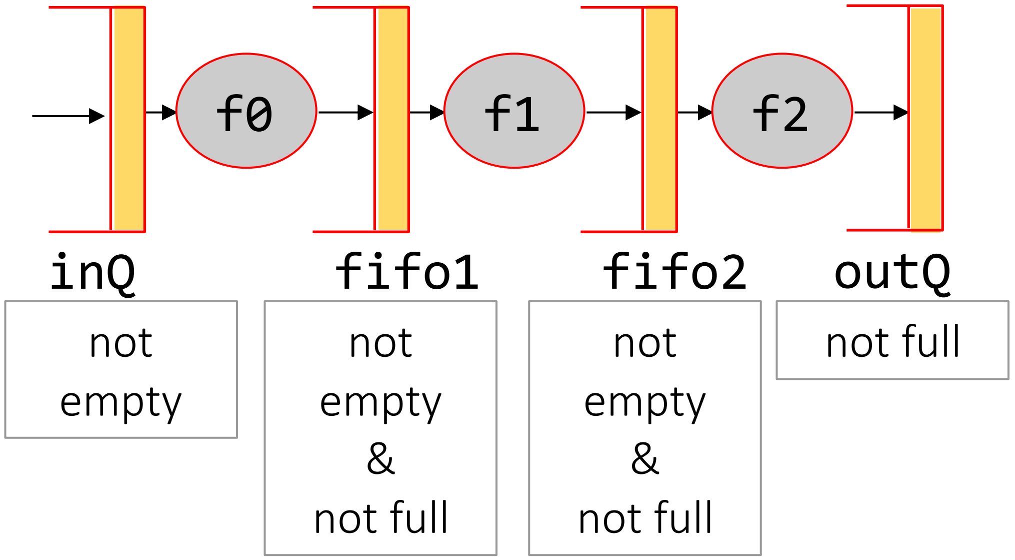
6.004 Recitation Problems
L21 – Implementing Pipelines

Problem 1: Complete the design below. (Note: Sfifo = Searchable Fifo)



```
module mkBypassRFile(BypassRFile);
  RFile    rf <- mkRFile;
  SFifo#(1, RIdxData#(RIndx, Data))
                bypass <- mkBypassSFifo;
  rule move;


    begin let x = bypass.first;
        rf[x.index] <= x.data;
        bypass.deq end;


  endrule

  method Action wr(RIndx rindx, Data data);

    if (rindx!=0) bypass.enq(
            RIdxData{index:rindx, data:data});

  endmethod

  method Data rd(RIndx rindx) =

      return (!bypass.search1(rindx)) ? rf.rd1(rindx)
             : bypass.read1(rindx);

endmodule

typedef struct {RIndx index; Data data}
RIdxData deriving (Bits);
```

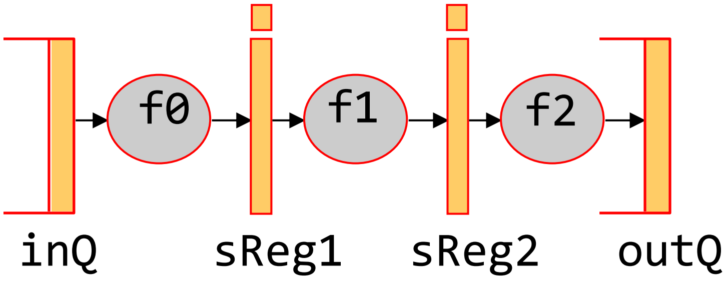
Problem 2: Can any pipeline stages fire concurrently if the FIFOs do not permit concurrent enq and deq?



Sol: Alternate stages in the pipeline can still fire concurrently

Problem 3: When is this rule enabled?

```
rule sync-pipeline;
 if (inQ.notEmpty)
  begin sReg1 <= f0(inQ.first);
inQ.deq;
        sReg1v <= Valid end
  else  sReg1v <= Invalid;
  sReg2 <= f1(sReg1); sReg2v <=
sReg1v;
 if (sReg2v == Valid)
outQ.enq(f2(sReg2));
endrule
```



inQ	sReg1v	sReg2v	outQ	
NE	V	V	NF	Yes
NE	V	V	F	No
NE	V	I	NF	Yes
NE	V	I	F	Yes
NE	I	V	NF	Yes
NE	I	V	F	No
NE	I	I	NF	Yes
NE	I	I	F	Yes

inQ	sReg1v	sReg2v	outQ	
E	V	V	NF	Yes
E	V	V	F	No
E	V	I	NF	Yes
E	V	I	F	Yes
E	I	V	NF	Yes
E	I	V	F	No
E	I	I	NF	Yes/NC
E	I	I	F	Yes

Limitations of registers

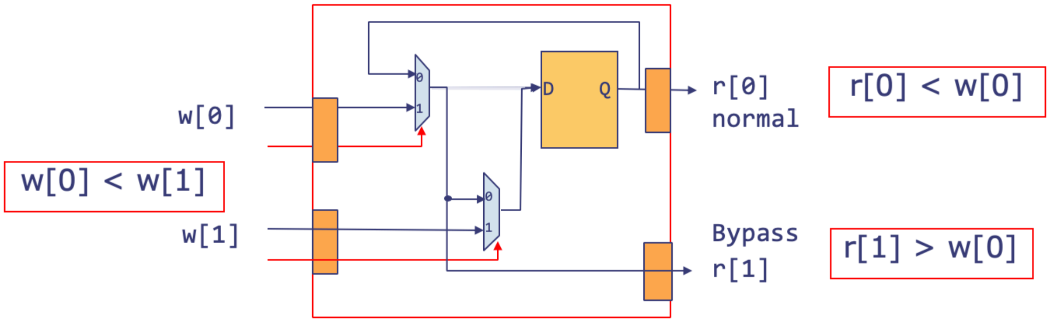
- Using the register primitive no *communication* can take place in the same clock cycle between
  - two methods or
  - two rules or
  - a rule and a method

EHRs to the rescue …

Ephemeral History Register (EHR): a primitive element to remedy this problem

Ephemeral History Register (EHR)

Dan Rosenband [MEMOCODE'04]



r[0] < w[0]

w[0] < w[1]

r[1] > w[0]

- r[1] returns:
  - the current state if w[0] *is not enabled*
  - the value being written if w[0] *is enabled*
- w[1] has higher priority than w[0]