```
┌──────────────────── MODULE WrongWriteThroughCache ────────────────────┐
EXTENDS Naturals, Sequences, MemoryInterface
VARIABLES wmem, ctl, buf, cache, memQ
CONSTANT QLen
ASSUME (QLen ∈ Nat) ∧ (QLen > 0)
M ≜ INSTANCE InternalMemory WITH mem ← wmem
├───────────────────────────────────────────────────────────────────────┤
Init ≜ ∧ M!IInit
       ∧ cache = [p ∈ Proc ↦ [a ∈ Adr ↦ NoVal]]
       ∧ memQ = ⟨⟩

TypeInvariant ≜
  ∧ wmem   ∈ [Adr → Val]
  ∧ ctl    ∈ [Proc → {"rdy", "busy", "waiting", "done"}]
  ∧ buf    ∈ [Proc → MReq ∪ Val ∪ {NoVal}]
  ∧ cache  ∈ [Proc → [Adr → Val ∪ {NoVal}]]
  ∧ memQ   ∈ Seq(Proc × MReq)

Coherence ≜ ∀ p, q ∈ Proc, a ∈ Adr :
               (NoVal ∉ {cache[p][a], cache[q][a]})
                   ⇒ (cache[p][a] = cache[q][a])
├───────────────────────────────────────────────────────────────────────┤
Req(p) ≜ M!Req(p) ∧ UNCHANGED ⟨cache, memQ⟩
Rsp(p) ≜ M!Rsp(p) ∧ UNCHANGED ⟨cache, memQ⟩

RdMiss(p) ≜  ∧ (ctl[p] = "busy") ∧ (buf[p].op = "Rd")
             ∧ cache[p][buf[p].adr] = NoVal
             ∧ Len(memQ) < QLen
             ∧ memQ′ = Append(memQ, ⟨p, buf[p]⟩)
             ∧ ctl′ = [ctl EXCEPT ![p] = "waiting"]
             ∧ UNCHANGED ⟨memInt, wmem, buf, cache⟩

DoRd(p) ≜
  ∧ ctl[p] ∈ {"busy", "waiting"}
  ∧ buf[p].op = "Rd"
  ∧ cache[p][buf[p].adr] ≠ NoVal
  ∧ buf′ = [buf EXCEPT ![p] = cache[p][buf[p].adr]]
  ∧ ctl′  = [ctl EXCEPT ![p]  = "done"]
  ∧ UNCHANGED ⟨memInt, wmem, cache, memQ⟩

DoWr(p) ≜
  LET r  ≜ buf[p]
  IN   ∧ (ctl[p] = "busy") ∧ (r.op = "Wr")
       ∧ Len(memQ) < QLen
       ∧ cache′ = [q ∈ Proc ↦
                       IF (p = q)  ∨  (cache[q][r.adr] ≠ NoVal)
```

```
                    THEN [cache[q] EXCEPT ![r.adr] = r.val]
                    ELSE  cache[q]]
        ∧ memQ′ = Append(memQ, ⟨p, r⟩)
        ∧ buf′ = [buf EXCEPT ![p] = NoVal]
        ∧ ctl′  = [ctl EXCEPT ![p]  = "done"]
        ∧ UNCHANGED ⟨memInt, wmem⟩

vmem  ≜
  LET f[i ∈ 0 . . Len(memQ)]  ≜
        IF i = 0 THEN wmem
                ELSE IF memQ[i][2].op = "Rd"
                       THEN f[i − 1]
                       ELSE  [f[i − 1] EXCEPT ![memQ[i][2].adr] =
                                                     memQ[i][2].val]
  IN   f[Len(memQ)]

MemQWr  ≜  LET r  ≜  Head(memQ)[2]
             IN   ∧ (memQ ≠ ⟨⟩)  ∧   (r.op = "Wr")
                  ∧ wmem′  = [wmem EXCEPT ![r.adr] = r.val]
                  ∧ memQ′  = Tail(memQ)
                  ∧ UNCHANGED ⟨memInt, buf, ctl, cache⟩

MemQRd  ≜
  LET p   ≜  Head(memQ)[1]
      r  ≜  Head(memQ)[2]
  IN    ∧ (memQ ≠ ⟨⟩) ∧ (r.op = "Rd")
        ∧ memQ′  = Tail(memQ)
        ∧ cache′ = [cache EXCEPT ![p][r.adr] = wmem[r.adr]]   Wrong: 'vmen' has been deliberately replaced by 'wme
        ∧ UNCHANGED ⟨memInt, wmem, buf, ctl⟩

Evict(p, a)  ≜  ∧ (ctl[p] = "waiting") ⇒ (buf[p].adr ≠ a)
                ∧ cache′ = [cache EXCEPT ![p][a] = NoVal]
                ∧ UNCHANGED ⟨memInt, wmem, buf, ctl, memQ⟩

Next  ≜  ∨ ∃ p ∈ Proc : ∨ Req(p) ∨ Rsp(p)
                        ∨ RdMiss(p) ∨ DoRd(p) ∨ DoWr(p)
                        ∨ ∃ a ∈ Adr : Evict(p, a)
         ∨ MemQWr ∨ MemQRd

Spec  ≜
  Init ∧ □[Next]_⟨memInt, wmem, buf, ctl, cache, memQ⟩
├──────────────────────────────────────────────────────────────┤
THEOREM Spec ⇒ □(TypeInvariant ∧ Coherence)
├──────────────────────────────────────────────────────────────┤
LM  ≜  INSTANCE Memory
THEOREM Spec ⇒ LM ! Spec
└──────────────────────────────────────────────────────────────┘
```