MODULE *simple_lock*

EXTENDS *Naturals*, *TLC*
CONSTANTS *NumReaders*, *NumWrites*

--algorithm *simple_lock*
variables $cur = 0$, $lock = 0$

process *writer* $= 1$
variable *write* $= 0$
begin
Writer_Loop:
  while $write \neq NumWrites$ **do**
    Write_Acquire:
      await $lock = 0$ **;**
      $lock := 1$ **;**
    Update:
      $cur := write$ **;**
    Write_Release:
      $lock := 0$ **;**
  end while ;
end process ;

process *reader* $\in 2 \,..\, (NumReaders + 1)$
variables *saved_read*
begin
Read_Acquire:
  await $lock = 0$ **;**
  $lock := 1$ **;**
Read:
  $saved\_read := cur$ **;**
Check:
  assert $saved\_read = cur$ **;**
Read_Release:
  $lock := 0$ **;**
end process ;

end algorithm ;
BEGIN TRANSLATION ($chksum(pcal) =$ "1359a2df" $\land\ chksum(tla) =$ "8136cae")
CONSTANT *defaultInitValue*
VARIABLES *cur*, *lock*, *pc*, *write*, *saved_read*

$vars \triangleq \langle cur, lock, pc, write, saved\_read \rangle$

$ProcSet \triangleq \{1\} \cup (2 \,..\, (NumReaders + 1))$

$Init \triangleq$ Global variables
  $\land\ cur = 0$

$\land\ lock = 0$

Process writer

$\land\ write = 0$

Process reader

$\land\ saved\_read = [self \in 2\,..\,(NumReaders + 1) \mapsto defaultInitValue]$

$\land\ pc = [self \in ProcSet \mapsto \text{CASE } self = 1 \to \text{“Writer\_Loop”}$

$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \square\ \ self \in 2\,..\,(NumReaders + 1) \to \text{“Read\_Acquire”}]$

$Writer\_Loop \triangleq$
$\land\ pc[1] = \text{“Writer\_Loop”}$
$\land\ \text{IF } write \neq NumWrites$
$\qquad \text{THEN } \land\ pc' = [pc \text{ EXCEPT } ![1] = \text{“Write\_Acquire”}]$
$\qquad \text{ELSE } \land\ pc' = [pc \text{ EXCEPT } ![1] = \text{“Done”}]$
$\land\ \text{UNCHANGED } \langle cur,\ lock,\ write,\ saved\_read\rangle$

$Write\_Acquire \triangleq$
$\land\ pc[1] = \text{“Write\_Acquire”}$
$\land\ lock = 0$
$\land\ lock' = 1$
$\land\ pc' = [pc \text{ EXCEPT } ![1] = \text{“Update”}]$
$\land\ \text{UNCHANGED } \langle cur,\ write,\ saved\_read\rangle$

$Update \triangleq$
$\land\ pc[1] = \text{“Update”}$
$\land\ cur' = write$
$\land\ pc' = [pc \text{ EXCEPT } ![1] = \text{“Write\_Release”}]$
$\land\ \text{UNCHANGED } \langle lock,\ write,\ saved\_read\rangle$

$Write\_Release \triangleq$
$\land\ pc[1] = \text{“Write\_Release”}$
$\land\ lock' = 0$
$\land\ pc' = [pc \text{ EXCEPT } ![1] = \text{“Writer\_Loop”}]$
$\land\ \text{UNCHANGED } \langle cur,\ write,\ saved\_read\rangle$

$writer \triangleq Writer\_Loop \lor Write\_Acquire \lor Update \lor Write\_Release$

$Read\_Acquire(self) \triangleq$
$\land\ pc[self] = \text{“Read\_Acquire”}$
$\land\ lock = 0$
$\land\ lock' = 1$
$\land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“Read”}]$
$\land\ \text{UNCHANGED } \langle cur,\ write,\ saved\_read\rangle$

$Read(self) \triangleq$
$\land\ pc[self] = \text{“Read”}$
$\land\ saved\_read' = [saved\_read \text{ EXCEPT } ![self] = cur]$
$\land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“Check”}]$
$\land\ \text{UNCHANGED } \langle cur,\ lock,\ write\rangle$

$Check(self) \triangleq$
$\land\ pc[self] = \text{“Check”}$
$\land\ Assert(saved\_read[self] = cur,$
$\qquad\qquad \text{“Failure of assertion at line 32, column 5.”})$
$\land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“Read\_Release”}]$

$\land$ UNCHANGED $\langle cur,\ lock,\ write,\ saved\_read\rangle$

$Read\_Release(self) \triangleq \land\ pc[self] = \text{“Read\_Release”}$
$\land\ lock' = 0$
$\land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“Done”}]$
$\land$ UNCHANGED $\langle cur,\ write,\ saved\_read\rangle$

$reader(self) \triangleq Read\_Acquire(self) \lor Read(self) \lor Check(self)$
$\lor\ Read\_Release(self)$

Allow infinite stuttering to prevent deadlock on termination.

$Terminating \triangleq \land \forall\, self \in ProcSet : pc[self] = \text{“Done”}$
$\land$ UNCHANGED $vars$

$Next \triangleq writer$
$\lor\ (\exists\, self \in 2\,..\,(NumReaders + 1) : reader(self))$
$\lor\ Terminating$

$Spec \triangleq Init \land \Box[Next]_{vars}$

$Termination \triangleq \Diamond(\forall\, self \in ProcSet : pc[self] = \text{“Done”})$

END TRANSLATION