int compareAndSwap(int locked, 0, 1); locked =0

{

int temp=locked;

if (temp==0)

locked=1;

return temp;

}

while( compareAndSwap(locked, 0,1)!=0)

/\*Enter Critical region\*/

/\*Exit critical region\*/ locked=0;

/\*execute remainder section of the code\*/

Bound Waiting

**TestAndSet**

Another hardware solution for IPC to enter the critical section. In TestAndSet, we have a shared lock variable which can take either of the two values, 0 or 1.

0-> Lock

1-> Unlock

Before entering into the critical section, a process enquires about the lock. If it is locked, it keeps on waiting till it becomes free and if it is not locked; it takes the lock and executes the critical section.

In TestAndSet, Mutual exclusion and progress are preserved but bounded waiting (There exists a bound, or limit, on the number of times other processes are allowed to enter their critical sections after a process has made request to enter its critical section and before that request is granted) cannot be preserved.

int TestAndSet(int &lock)

{

int initial = lock;

lock = 1;

return initial;

}

void enter\_CS(X)

{

while test-and-set(X) ;

lock =0;

}

void leave\_CS(X)

{ lock =1;

}