Note on Lab 4

Yuji Kukimoto

May 16, 1996

It turned out that the exclusive property holds on `ex1a.v` although I claimed that it did not. The controller module in `ex1a.v` still has a bug as you fixed it yesterday in Lab 6 (The `pass_token` signal is not reset to zero for the transition edge from IDLE to READY.), but the behavior of the client module is so restricted that the bug is not observable by just looking at input/output behaviors.

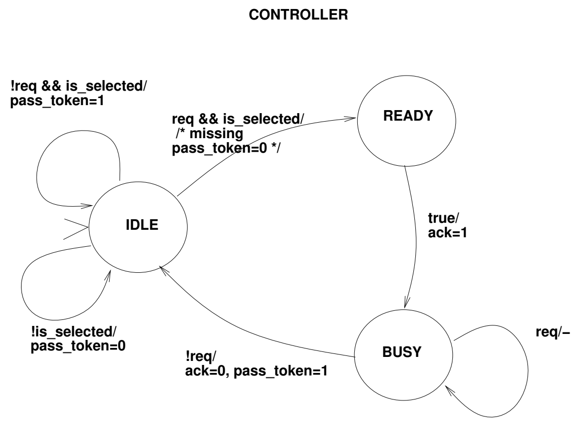
Let us think about why we cannot manifest the bug in the controller module. To generate an erroneous token while the bus is used by someone else, the following transitions must be taken.

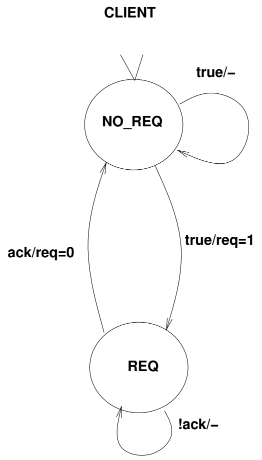
> The controller of a client leaves the state BUSY for the state IDLE, and at the next clock goes to the state READY. The first transition sets the variable `pass_token` to one, and the second transition fails to reset the variable, thereby generating a token although the client is in the READY state.

Let us try to see if we can activate these transitions. In the following, we focus on clientA. When the controller is ready to leave the state BUSY, the arbiter is always selecting A (See below). Therefore, when the controller moves from the state BUSY to the state IDLE, the arbiter points to B. Because A cannot be selected at that time, it is impossible to take the edge from the state IDLE to the state READY. Therefore the design bug in the controller does not invalidate the exclusive property.

```
shambhala% cat ex1a.ctl
AG ( controllerA.state=BUSY * reqA=0 -> arbiter.state=A );

EF ( controllerA.state=BUSY * EX( controllerA.state=IDLE * EX( controllerA.state
=READY)));

AG ( !(ackA = 1 * ackB = 1 + ackB = 1 * ackC = 1 + ackC = 1 * ackA=1) );

shambhala% /projects/vis/vis-devel/mips-o/vis
vis release 1.1 (compiled 2-Apr-96 at 9:16 PM)
```

```
Sourcing .visrc of Tom
vis> rlmv ex1a.mv
Warning: Model main may have a cyclic connection which involves variable ackA
vis> init
vis> mc ex1a.ctl

# MC: formula passed --- AG(((controllerA.state=BUSY * reqA=0) -> arbiter.state=
A))
# MC: formula failed --- EF((controllerA.state=BUSY * EX((controllerA.state=IDLE
 * EX(controllerA.state=READY)))))

# MC: Calling debugger


--State
ackA:0
ackB:0
ackC:0
arbiter.state:A
clientA.state:NO_REQ
clientB.state:NO_REQ
clientC.state:NO_REQ
controllerA.state:IDLE
controllerB.state:IDLE
controllerC.state:IDLE
pass_tokenA:1
pass_tokenB:1
pass_tokenC:1
reqA:0
reqB:0
reqC:0

fails EF((controllerA.state=BUSY * EX((controllerA.state=IDLE * EX(controllerA.s
tate=READY)))))

--A simple counter example does not exist since it
  requires generating all paths from the state

# MC: formula passed --- AG(!((((ackA=1 * ackB=1) + (ackB=1 * ackC=1)) + (ackC=1
 * ackA=1))))
```





Notation:
input condition / output activity

NOTE: ack and pass_token are registers

ack=1 on an edge means that ack gets 1 at the next clock
– in output activity means no activity