Appendix

SCSI Timing Diagrams

Timing diagrams are commonly used in electrical engineering to show values of electrical wires with respect to time. The pulse-like lines in the timing diagram indicate the value of electrical wires. In this section, a wire is said to be inactive if the value of the pulse is 1, i.e. high. Otherwise, a wire is said to be active.

Figure 1 shows devices 2 and 6 trying to compete for the bus, with device 6 being the winner. Device 6 then selects device 5 as the target device. This figure will be explained in detail to enhance the understanding of the timing diagrams as well as the SCSI protocol. The numbers in front of each explanation correspond to the numbers in boldface in Figure 1. A similar explanation of the SCSI protocol can also be found in section 5.

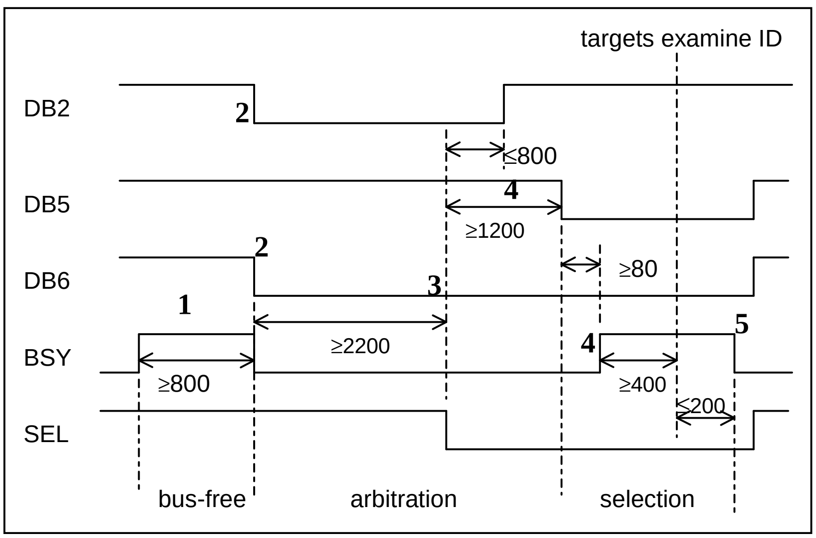

Figure 1. Example timing diagram on SCSI bus

1. At the beginning, there is a period where both *BSY* and *SEL* are inactive. That period has to be longer than 800ns in order to be qualified as the *bus-free* phase. Otherwise, devices on the bus should not consider that short period as the *bus-free* phase and should just ignore it. These kinds of strict timing constraints apply throughout the whole protocol.
2. When the bus is in the *bus-free* phase, devices interested in obtaining the bus should activate their corresponding data lines. *BSY* is also be activated to indicate the start of the *arbitration* phase. In Figure 1, the data lines for device 2 and 6 are activated indicating their intentions.

3. After waiting for 2200ns to ensure that all devices have a chance to activate their data lines, all devices should examine all data lines and determine the winner, and they should obtain the same result independently. Device 6, who is the winner, activates *SEL*.
4. Losers must deactivate their data lines within 800ns, and the winner activates the data line of the target device after 1200ns. After waiting for another 80ns to allow for electrical delays, *BSY* is deactivated to indicate the start of the *selection* phase.
5. 400ns after that, all devices examine their data lines to check if they have been selected. The selected target responds by activating *BSY* within 200ns. Finally, the winner deactivates *SEL* and all data lines it has activated.

Figure 2 shows the timing diagram for the scenario where device 5 reselects device 6 as the target device.

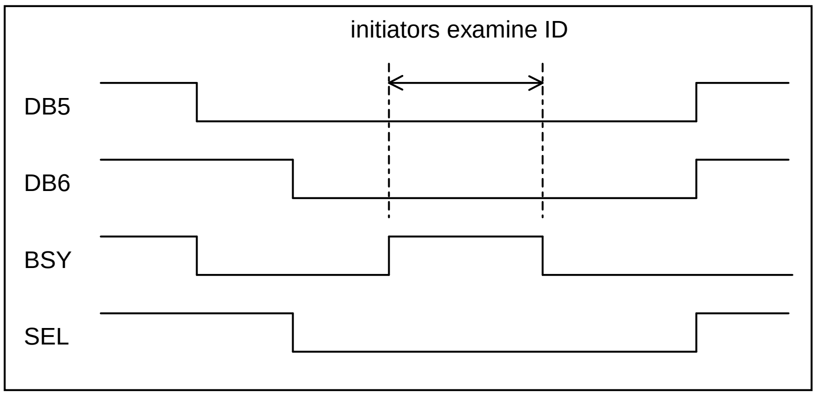

Figure 2. Simplified timing diagram for the reselection

SCSI Example in UPPAAL Graphical Format

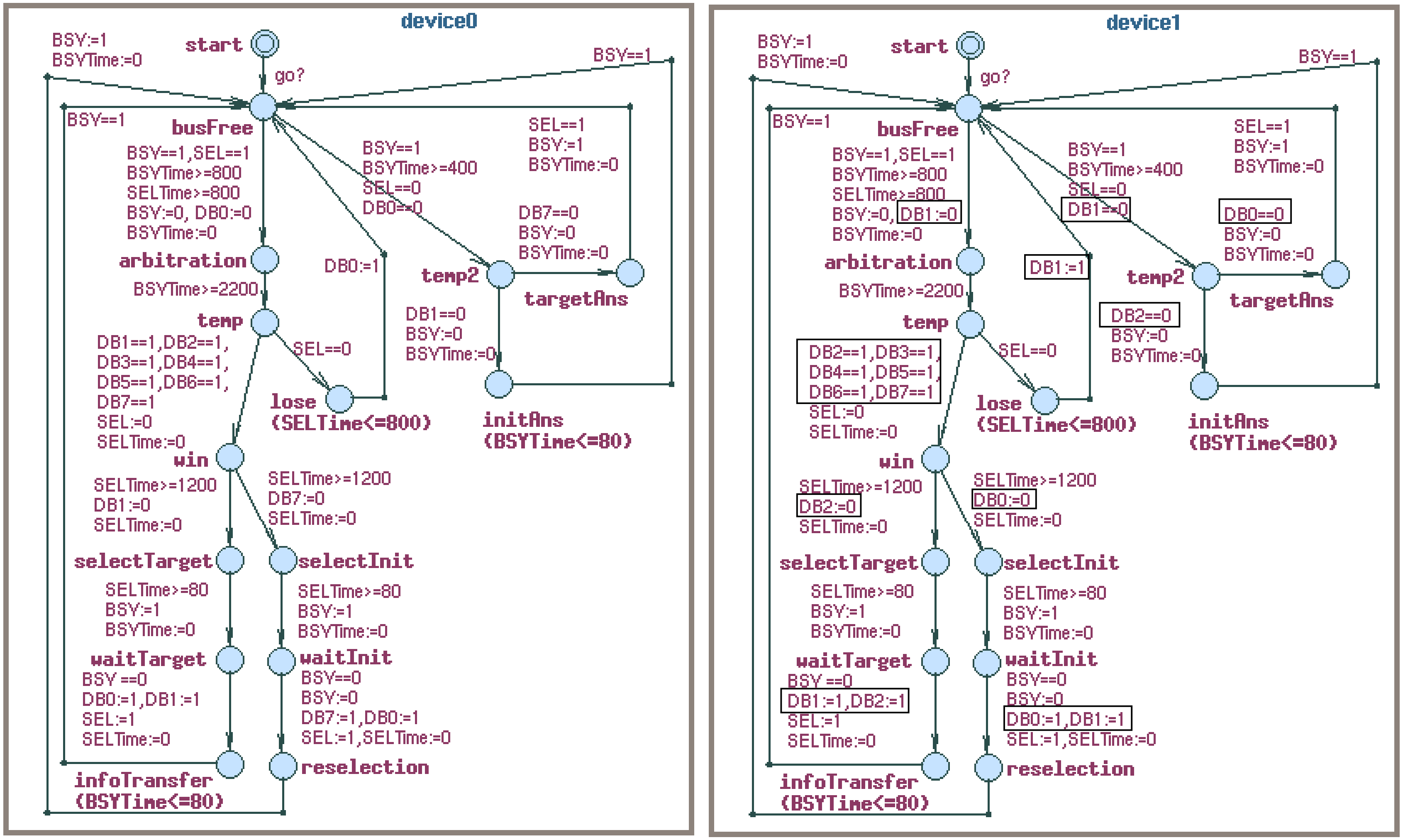

Figure 3. Devices 0 and 1 in the SCSI protocol example with differences enclosed in rectangles

Devices 0 and 1 in the SCSI protocol example are modeled in the UPPAAL modeling language and are shown in Figure 3. It can be seen that both automata are similar to each other. Therefore, the rest of the 6 devices in the SCSI protocol example are inferred by showing the differences between them. The differences are enclosed in rectangles and are explained below :

Let x be an integer, $0 \le x \le 7$.

- For the transition between the state *busFree* and the state *arbitration*, *DB1:=0* activates the data line of device 1. For the automaton of device *x*, it will be *DBx :=0.*
- For the transition between the state *temp* and the state *win*, all data lines except the one for devices 0 and 1 are checked. For the automaton of device *x*, it will be *DBy==1*, for *y > x*.
- For the transition between the state *win* and the state *selectTarget*, *DB2:=0* activates the data line of the target device. For the automaton of device *x*, it will be *DBt :=0, t = x+1 mod 8.*
- For the transition between the state *waitTarget* and the state *infoTransfer*, *DB1:=1* and *DB2:=1* deactivate the data lines of the device 1 and its target device. For the automaton of device *x*, it will be *DBx:=1* and *DBt :=1, t = x+1 mod 8.*
- For the transition between the state *win* and the state *selectInit*, *DB0:=0* activates the data line of the initiator device. For the automaton of device *x*, it will be *DBi :=0, i = x-1 mod 8.*
- For the transition between the state *waitInit* and the state *reselection*, *DB0:=1* and *DB1:=1* deactivate the data lines of the device 1 and its initiator device. For the automaton of device *x*, it will be *DBx:=1* and *DBi :=1, i = x-1 mod 8.*
- For the transition between the state *lose* and the state *busFree*, *DB1:=1 de*activates the data line of the device 1. For the automaton of device *x*, it will be *DBx:=1*.
- For the transition between the state *busFree* and the state *temp2*, *DB1:=0* activates the data line of device 1. For the automaton of device *x*, it will be *DBx :=0*.
- For the transition between the state *temp2* and the state *initAns*, *DB2==0* checks the data line of the target device. For the automaton of device *x*, it will be *DBt==0, t = x+1 mod 8.*
- For the transition between the state *temp2* and the state *targetAns*, *DB0==0* checks the data line of the initiator device. For the automaton of device *x*, it will be *DBi==0, i = x-1 mod 8.*

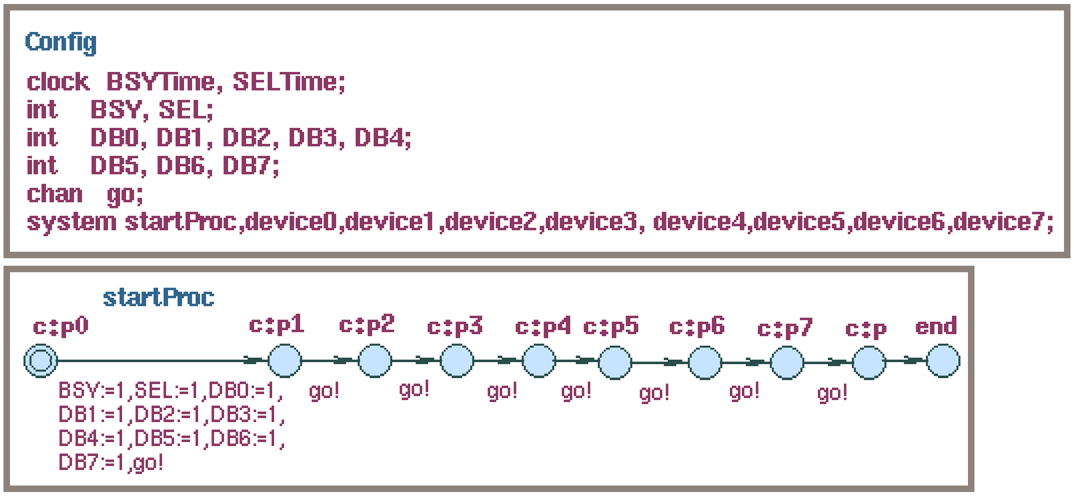

Figure 4. Config Box and the initialization process in the SCSI Protocol Example

At the top of Figure 4 shows the *Config* box that declares all variables, channels and automata used in the whole model. The bottom box shows the initialization process that performs all necessary initializations.

The textual format of the SCSI protocol model can be automatically generated by UPPAAL and will not be included in this report. However, interested readers can obtain the information at http://www.cs.toronto.edu/~andre/scsi.txt.

UPPAAL Properties of SCSI Protocol

```
/////////////////////////////////////////////////////////////
// A[](deviceX.win -> not(conjunction of all deviceY.win's,
// Y!=X), X,Y=0..7
// there can only be one winner at any point in time
/////////////////////////////////////////////////////////////
A[] (device0.win imply not (device1.win or device2.win or
device3.win or device4.win or device5.win or device6.win or
device7.win) )
A[] (device1.win imply not (device0.win or device2.win or
device3.win or device4.win or device5.win or device6.win or
device7.win) )
A[] (device2.win imply not (device0.win or device1.win or
device3.win or device4.win or device5.win or device6.win or
device7.win) )
A[] (device3.win imply not (device0.win or device1.win or
device2.win or device4.win or device5.win or device6.win or
device7.win) )
A[] (device4.win imply not (device0.win or device1.win or
device2.win or device3.win or device5.win or device6.win or
device7.win) )
A[] (device5.win imply not (device0.win or device1.win or
device2.win or device3.win or device4.win or device6.win or
device7.win) )
A[] (device6.win imply not (device0.win or device1.win or
device2.win or device3.win or device4.win or device5.win or
device7.win) )
A[] (device7.win imply not (device0.win or device1.win or
device2.win or device3.win or device4.win or device5.win or
device6.win) )


/////////////////////////////////////////////////////////////
// everybody has a chance to grab the bus
/////////////////////////////////////////////////////////////
E<> (device0.win)
E<> (device1.win)
E<> (device2.win)
E<> (device3.win)
E<> (device4.win)
E<> (device5.win)
E<> (device6.win)
E<> (device7.win)
```

```
/////////////////////////////////////////////////////////////
// normally in the selection phase, an initiator device
// selects a target device and the target device answers
// back to the initiator device. Therefore, it is
// impossible for a target device to answer the selection
// if no initiator device exists
/////////////////////////////////////////////////////////////

A[] ((device0.targetAns and SEL==0) imply
      device7.waitTarget)
A[] ((device1.targetAns and SEL==0) imply
      device0.waitTarget)
A[] ((device2.targetAns and SEL==0) imply
      device1.waitTarget)
A[] ((device3.targetAns and SEL==0) imply
      device2.waitTarget)
A[] ((device4.targetAns and SEL==0) imply
      device3.waitTarget)
A[] ((device5.targetAns and SEL==0) imply
      device4.waitTarget)
A[] ((device6.targetAns and SEL==0) imply
      device5.waitTarget)
A[] ((device7.targetAns and SEL==0) imply
      device6.waitTarget)


/////////////////////////////////////////////////////////////
// normally in the reselection phase, a target device
// reselects an initiator device and the initiator device
// answers back to the target device. Therefore, it is
// impossible for an initiator device to answer the
// reselection if no target device exists
/////////////////////////////////////////////////////////////
A[] ((device0.initAns and SEL==0) imply device1.waitInit)
A[] ((device1.initAns and SEL==0) imply device2.waitInit)
A[] ((device2.initAns and SEL==0) imply device3.waitInit)
A[] ((device3.initAns and SEL==0) imply device4.waitInit)
A[] ((device4.initAns and SEL==0) imply device5.waitInit)
A[] ((device5.initAns and SEL==0) imply device6.waitInit)
A[] ((device6.initAns and SEL==0) imply device7.waitInit)
A[] ((device7.initAns and SEL==0) imply device0.waitInit)


/////////////////////////////////////////////////////////////
// all devices must be in idle state within 80 ns releasing
// BSY and SEL also, all devices must wait for SEL and BSY
// to settle for 800 ns before entering the arbitration
/////////////////////////////////////////////////////////////
A[] ((SEL==1 and BSY==1 and BSYTime>80 and BSYTime<800 and
SELTime>80 and SELTime<800) imply (device0.busFree and
device1.busFree and device2.busFree and device3.busFree and
device4.busFree and device5.busFree and device6.busFree and
device7.busFree ))
```

```
//////////////////////////////////////////////////////////////
// when two devices are transferring data, they must be in
// information transfer phase, indicated by SEL==1 and
// BSY==0
//////////////////////////////////////////////////////////////

A[]((device0.infoTransfer and device1.targetAns) imply
    (SEL==1 and BSY==0))
A[]((device0.reselection and device7.initAns) imply
    (SEL==1 and BSY==0))
A[]((device1.infoTransfer and device2.targetAns) imply
    (SEL==1 and BSY==0))
A[]((device1.reselection and device0.initAns) imply
    (SEL==1 and BSY==0))
A[]((device3.infoTransfer and device4.targetAns) imply
    (SEL==1 and BSY==0))
A[]((device3.reselection and device2.initAns) imply
    (SEL==1 and BSY==0))
A[]((device4.infoTransfer and device5.targetAns) imply
    (SEL==1 and BSY==0))
A[]((device5.reselection and device3.initAns) imply
    (SEL==1 and BSY==0))
A[]((device5.infoTransfer and device6.targetAns) imply
    (SEL==1 and BSY==0))
A[]((device5.reselection and device4.initAns) imply
    (SEL==1 and BSY==0))
A[]((device6.infoTransfer and device7.targetAns) imply
    (SEL==1 and BSY==0))
A[]((device6.reselection and device5.initAns) imply
    (SEL==1 and BSY==0))
A[]((device7.infoTransfer and device0.targetAns) imply
    (SEL==1 and BSY==0))
A[]((device7.reselection and device6.initAns) imply
    (SEL==1 and BSY==0))


//////////////////////////////////////////////////////////////
// while a device is in selection state and before another
// device answers, SEL==0 and BSY==1
//////////////////////////////////////////////////////////////
A[]((device0.waitTarget and not device1.targetAns) imply
    (SEL==0 and BSY==1))
A[]((device0.waitInit and not device7.initAns) imply
    (SEL==0 and BSY==1))
A[]((device1.waitTarget and not device2.targetAns) imply
    (SEL==0 and BSY==1))
A[]((device1.waitInit and not device0.initAns) imply
    (SEL==0 and BSY==1))
A[]((device2.waitTarget and not device3.targetAns) imply
    (SEL==0 and BSY==1))
A[]((device2.waitInit and not device1.initAns) imply
    (SEL==0 and BSY==1))
A[]((device3.waitTarget and not device4.targetAns) imply
    (SEL==0 and BSY==1))
A[]((device3.waitInit and not device2.initAns) imply
    (SEL==0 and BSY==1))
A[]((device4.waitTarget and not device5.targetAns) imply
    (SEL==0 and BSY==1))
A[]((device4.waitInit and not device3.initAns) imply
    (SEL==0 and BSY==1))
A[]((device5.waitTarget and not device6.targetAns) imply
    (SEL==0 and BSY==1))
A[]((device5.waitInit and not device4.initAns) imply
    (SEL==0 and BSY==1))
A[]((device6.waitTarget and not device7.targetAns) imply
    (SEL==0 and BSY==1))
A[]((device6.waitInit and not device5.initAns) imply
    (SEL==0 and BSY==1))
A[]((device7.waitTarget and not device0.targetAns) imply
    (SEL==0 and BSY==1))
A[]((device7.waitInit and not device6.initAns) imply
    (SEL==0 and BSY==1))


//////////////////////////////////////////////////////////////
// SEL==1 and BSY==0 while a device is in arbitration phase
//////////////////////////////////////////////////////////////
A[] (device0.arbitration imply (SEL==1 and BSY==0))
A[] (device1.arbitration imply (SEL==1 and BSY==0))
A[] (device2.arbitration imply (SEL==1 and BSY==0))
A[] (device3.arbitration imply (SEL==1 and BSY==0))
A[] (device4.arbitration imply (SEL==1 and BSY==0))
A[] (device5.arbitration imply (SEL==1 and BSY==0))
A[] (device6.arbitration imply (SEL==1 and BSY==0))
A[] (device7.arbitration imply (SEL==1 and BSY==0))

//////////////////////////////////////////////////////////////
// (Reduced) Output from the UPPAAL verifier
// The second last line shows that the whole process
// took about 4 seconds
//////////////////////////////////////////////////////////////
Property 1 (line 1) is satisfied.
Property 2 (line 2) is satisfied.
...
...
Property 70 (line 90) is satisfied.
Property 71 (line 91) is satisfied.
2.88user     0.02system     0:04.03elapsed     71%CPU
(0avgtext+0avgdata 0,axresident)k
0inputs+0outputs (190major+89minor)pagefaults 0swaps
```