Prof. Dr. A. Podelski, Dr. B. Westphal
S. Andisha

Sommersemester 2012

Real-Time Systems

http://swt.informatik.uni-freiburg.de/teaching/SS2012/rtsys

Exercise Sheet 6

Early submission: Monday, **2012-07-23**, 12:00 Regular submission: Tuesday, **2012-07-24** (!), 12:00

Exercise 1: SPI Bus [1] (20/20 Points + 20 Bonus)

The *Serial Peripheral Interface Bus* or SPI bus is an inter-chip communication standard that enables components on an electronic circuit to exchange information with relatively high data throughput while using as few hardware lines as possible.
The SPI Bus implements full-duplex point-to-point communication by using separate wires in each direction called $MISO$ (Master Input Slave Output) and $MOSI$ (Master Output Slave Input).
Communication in the bus is controlled by a device designed as "master". It is responsible for enabling bus transfers to and from individual "slave" devices by activating a dedicated *chip select* line (CS) exclusive for each slave. Thus, the master must therefore ensure the electrical integrity of the bus by never enabling more than one device at any given time. The master also keeps the serial bit transfers synchronized by exclusively driving a dedicated clock line ($SCLK$).

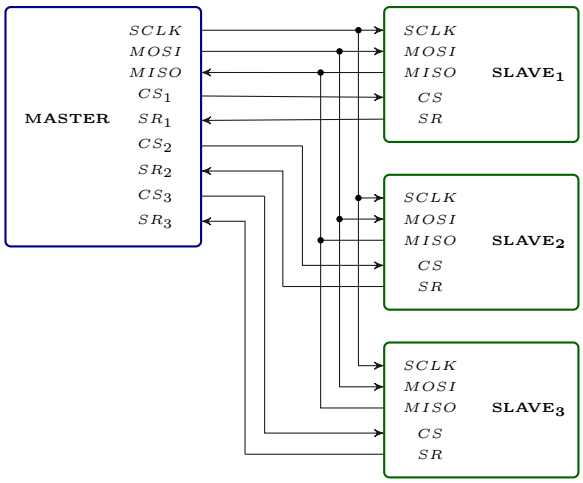

Figure 1: Schematic of a typical SPI Bus configuration with 3 slave devices.

Whenever the master wants to initiate a transfer, it activates the CS line of the corresponding slave and then starts driving the clock line to signal the individual bits. The duration of a data transfer is variable and depends on the amount of data to be exchanged.
In some systems, asynchronous slave communication is enabled by using an additional dedicated exclusive *slave request* (SR) line. Whenever data is ready to transfer at a slave, it makes a request to the master by activating its SR line, the master will then initiate the transfer by using the usual CS mechanism at any later time.
Any reasonable implementation of the SPI bus standard with asynchronous slave communication should ensure that whenever a slave signals that it has data ready to transfer, it should never be left to starve, i.e., it should always get its request eventually serviced.

(i) Formalize the requirements for an SPI Bus System with asynchronous slave communication as described above using DC and explain them briefly.

- At least (a) the mutual communication exclusiveness and (b) the no-starvation requirements should be formalized.

- Any other requirements you deem necessary should be formalized as well.
- For every requirement briefly state whether it describes a safety, liveness or duration property and why. (3/20)

(ii) Provide an Uppaal model of a design of an appropriate SPI Bus master and 3 slave devices.

Assumptions:

- Assume data transfers take between 150μs and 850 μs.
- Assume data are available at non-deterministic intervals.

a) Make sure you prove that your model is reasonable by showing it does not contain deadlocks. 1/20

b) Provide simulation traces or queries that show that in your model successful data transfers actually take place. 1/20

c) Determine the maximal and miminal time it takes to transfer *two* different bunches of data from a slave to the master. 3/20

Hint: you may use Uppaal to prove these times.

d) Do the maximal/minimal times depend on the number of slaves? If no, please argue why not, if yes, provide formulae for min/max times parameterised in number of slaves. 5 Bonus

e) *Hint: For the subtasks (iia) to (iid), you may use the Uppaal model available from the lecture's homepage. The design follows a Time Division Multiple Access (TDMA) approach. Time is divided into* frames *of length* 6000μs *(starting at time 0), each frame is divided into three* slots *of length* 2000μs*, each slot in turn into two* windows *of length* 1000μs*. The first slot of each frame is assigned to slave 1, the second slot to slave 2, the third slot to slave 3.*

The first window of each slot is used for transferring data from master to slave, this is indicated by activating the corresponding CS line. In the second window, the CS line is activated again in a handshake window of length 50μs*, if the slave has data ready, it activates the SR line within additional* 50μs*, transfer from slave to master starts immediately afterwards.*

If you provide an Uppaal model of an *own* design of an appropriate SPI Bus master and 3 slave devices and use it for subtasks (iia) to (iid): 10/20

f) If your design significantly improves the maximal time it takes to transfer two different bunches of data from a slave to the master: 10 Bonus

(iii) Use the Uppaal model checker to show that the requirements are fulfilled by your design. 2/20

(iv) The Uppaal model of the TDMA approach provided on the homepage is meant to be parameterised in the number of slaves, i.e., it is supposed to be instantiable not only with 3 but with any number of slaves.

For absence of deadlocks and for one of the queries from the previous task provide a plot of verification time over number of slaves in that model.

How does the graph look? Is this coincidence? 5 Bonus

References

[1] Freescale Semiconductor. *Motorola M68HC11 Reference Manual - Rev. 6.1.* 2002.