SWS: Data Multiplexer

Eric Bouwers
Gideon Smeding
Gerbo Engels

4th October 2005

Part I: Transmission Protocol

$type\ Msg\ :=\ \mathrm{OK?} \mid \mathrm{YES} \mid \mathrm{NO} \mid \mathrm{DATA}\ d$

For the DC holds:

1. $\Box((asked \wedge in = \mathrm{NO} \wedge \mathrm{DATA}\ d) \Rightarrow \Diamond(out = \mathrm{OK?}))$
2. $\Box((asked \wedge in = \mathrm{YES} \wedge \mathrm{DATA}\ d) \Rightarrow \bigcirc(out = \mathrm{DATA}\ d \wedge \neg asked))$
3. $\Box((\neg asked \wedge \mathrm{DATA}\ d) \Rightarrow \bigcirc(out = \mathrm{OK?} \wedge asked))$

For the DP holds:

1. $\Box((in = \mathrm{OK?} \wedge ready) \Rightarrow \bigcirc(out = \mathrm{YES}))$
2. $\Box((in = \mathrm{OK?} \wedge \neg ready) \Rightarrow \bigcirc(out = \mathrm{NO}))$
3. $\Box(in = \mathrm{DATA}\ d \Rightarrow \bigcirc(\neg ready))$
4. $\Box(\neg ready \Rightarrow \Diamond(ready))$

Where *ready* is an internal variable of DP indicating whether the DP is ready or not.

Part II: BM Specification, some definitions

Constants

- *maxSlots* is the number data items the buffer can hold. *maxSlots* = 2 for this Buffer Multiplexer; it could be more than 2

Variables

- *freeSlots* is the number of free (unreserved) slots.
- *filledSlots* is the number of slots that are filled with data.

Channels

- DC.in = input of BM *from* a DC.
- DC.out = output of BM *to* a DC.
- DP.in = input of BM *from* the DP.
- DP.out = output of BM *to* the DP.

Part II: BM Specification

For all existing DC's (not necessary just two) holds:
Rules handling incoming OK? requests from DC:

1. $\Box(freeSlots = 0 \wedge DC.in = \text{OK?} \Rightarrow \bigcirc(DC.out = \text{NO}))$
2. $\Box(freeSlots > 0 \wedge DC.in = \text{OK?} \Rightarrow$
   $\bigcirc(freeSlots = freeSlots - 1 \wedge DC.out = \text{YES}))$
3. $\Box(DC.in = \text{OK?} \Rightarrow \Diamond(DC.out = \text{YES}))$

Buffering incoming DATA from DC:

4. $\Box(filledSlots = 1 \wedge DC.in = \text{DATA } d \Rightarrow$
   $\bigcirc(filledSlots = filledSlots + 1 \wedge slot = d \wedge DP.out = \text{OK?}))$
5. $\Box(filledSlots > 1 \wedge DC.in = \text{DATA } d \Rightarrow$
   $\bigcirc(filledSlots = filledSlots + 1 \wedge slot = d))$
   where $slot$ = any free slot in BM.

Part II: BM Specification

Rules handling anwsers from DP:

6. $\Box(DP.in = \text{NO} \Rightarrow \Diamond(DP.out = \text{OK?}))$
7. $\Box(filledSlots = 1 \wedge DP.in = \text{YES} \Rightarrow$
   $\bigcirc(filledSlots = 0 \wedge freeSlots = freeSlots + 1 \wedge DP.out = data))$
   where $data$ = the DATA from the filled slot in BM.
8. $\Box(filledSlots > 1 \wedge DP.in = \text{YES} \Rightarrow$
   $\bigcirc(filledSlots = filledSlots - 1 \wedge freeSlots = freeSlots + 1$
   $\wedge DP.out = data \wedge DP.out = \text{OK?}))$
   where $data$ = the DATA from any filled slot in BM.

Part II: BM Specification

Initial state:

9. $\Box(t = 0 \Rightarrow freeSlots = maxSlots \wedge filledSlots = 0)$

Ensure all data in the buffer will eventually be sent:

10. $\forall slot, data : \Box(slot.contains(data) \Rightarrow \Diamond(DP.out = data))$
    where $data$ :: DATA and $slot$ = a slot in BM.

Part III: Fairness of BM

The problem of fairness consists of two parts:

- All data from the DataCollectors should eventually enter the buffer
- All data from the buffer should eventually be sent to DataProcessor

This combined makes sure that all data from the DataCollectors will be processed.
The first part is specified by rule 3:

3. $\Box(DC.in = \text{OK?} \Rightarrow \Diamond(DC.out = \text{YES}))$

The second part is specified by rule 10:

10. $\forall slot, data : \Box(slot.contains(data) \Rightarrow \Diamond(DP.out = data))$
    where $data$ :: DATA and $slot$ = a slot in BM.