MODULE *Ass2 < YOURLOGIN >*

The Alternating Bit Protocol https://*en.wikipedia.org*/wiki/*Alternating_bit_protocol*
All code should contain brief documentation and be easy to understand.

Rename and finish this module. To add other modules > File > Open Module > Add TLA+ Module
Build the *dataWire* Module containing a FAULTY Channel with input and output - it can non deterministically send, corrupt or drop a packet.
This must be instantiated twice, *dataWire* for sending data and *ackWire* for sending acknowledgements.

Build two other Modules *abpSender* and *abpReceiver*.

The *abpSender* shares *dataSend* (a sequence of data items + bits) with the *dataWire* input and *ackSend* (a sequence of acknowlagement bits) with the *ackWire* output

The *abpReceiverr* shares *dataReceive* (a sequence of data items + bits) with the *dataWire* output and *ackReceive* (a sequence of acknowledgement bits) with the *ackWire* input

Build a one place buffer in the module *OnePlaceBufferSpec* and show that *abpMain* is a refinement of *OnePlaceBufferSpec*.

Complete the 6 parts below

18% (1) RENAME and finish this module (must be TLA+)
18% = 9% how thoroughly the protocol has been tested + 9% correctness
(parts 2 to 5 - 18% = 8% simplicity and inteligibility + 10% correctness)
18% (2) build *abpSender* (must contain *PlusCal*)
18% (3) build *abpReceiver* (must contain *PlusCal*)
18% (4) build *onePlaceBuffer* (may be TLA+ or *PluaCal*)
18% (5) build *dataWire* (must contain *PlusCal*)
10% (6) in this module explain what model you built and what they verify and explain what you have not verified
100%

HANDIN the zipped directory built from the TLA+ tool it must contain all 5 files + the models you used

The TLA+ code in this file is only gor guidence you are free to amend as you want.

EXTENDS *Naturals*, *Integers*, *TLC*, *Sequences*, *Bags*, *FiniteSets*
CONSTANT *CORRUPT_DATA*, *Message* to be set in model to a sequence
VARIABLES *sendDataQueue*, send data to *dataWire*
receiveDataQueue, receive data from *dataWire*
sendAckQueue, receive acknowlage from *ackWire*
receiveAckQueue, send acknowlage to *ackWire*
mess, message input to sender process
messOut, message output from receiver process
sender data
sender_bit,
*pc*1,

receiver data

rec_bit,
*pc*2

VARIABLES *read_state* ghost variable

$vars \triangleq \langle \quad \rangle$

. . . . . .
. . . . . .
dataWir $\triangleq$ INSTANCE *dataWire* WITH *inputW* $\leftarrow$ *sendDataQueue*, *outputW* $\leftarrow$ *receiveDataQueue*
ackWir $\triangleq$ INSTANCE *dataWire* WITH *inputW* $\leftarrow$ *receiveAckQueue*, *outputW* $\leftarrow$ *sendAckQueue*

Used for refinement check
OPB $\triangleq$ INSTANCE *OnePlaceBufferSpec*

---

ghost variables must not change the behaviour of the module. They are only used to define the refinement

Init $\triangleq$ $\land$ *read_state* = TRUE ghost variable only used in refinement
. . . . . .

dataChannel $\triangleq$ $\land$ *dataWir*!*Next*
$\land$ UNCHANGED $\langle$ . . . . $\rangle$

ackChannel $\triangleq$ $\land$ *ackWir*!*Next*
$\land$ UNCHANGED $\langle$ . . . . $\rangle$

. . . .

Next $\triangleq$ *dataChannel* $\lor$ *ackChannel* $\lor$ . . . . . .

Spec $\triangleq$ *Init* $\land \Box[Next]_{vars}$

Add invariants + properties + Explain to a non expert what they show.

Your explanation (10%)
Briefly describe the models you used and what they verify

Explain what has not been verified.

---

* Modification History
* Last modified *Mon Apr* 29 08:21:36 *NZST* 2019 by *dstr*
* Created *Thu Feb* 14 15:38:49 *NZDT* 2019 by *dstr*