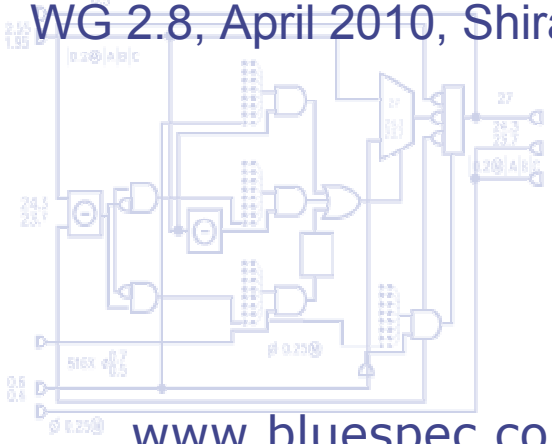
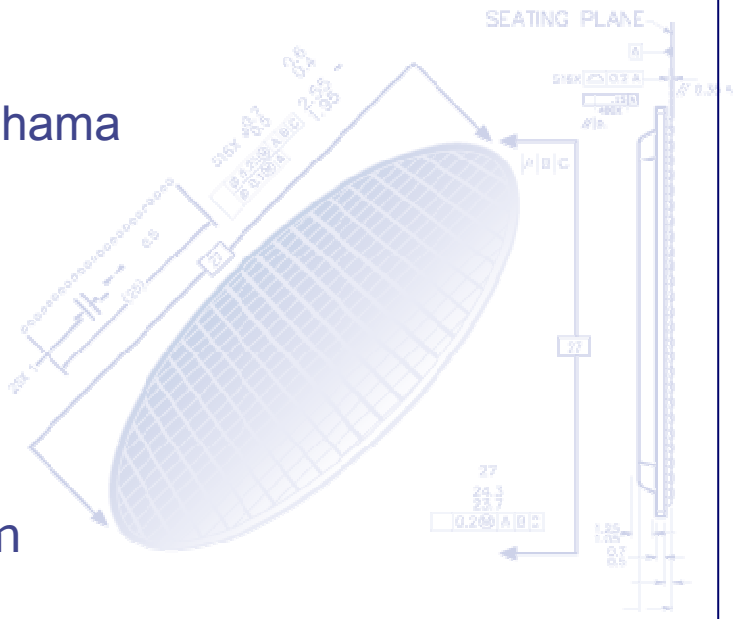
A programming/specification and verification problem based on the Intel QPI protocol ("QuickPath Interconnect")

Rishiyur S. Nikhil

WG 2.8, April 2010, Shirahama

```
import FIFO::*;

typedef Bit#(32) DataT;

module ex_in1_out2_bs(Empty);

  Integer fifo_depth = 16;

  function Bit#(1)  determine_queue(DataT val );
    return (val[0]);
  endfunction

  FIFO#(DataT) inbound1();
  mkSizedFIFO#(fifo_depth) the_inbound1(inbound1);
  FIFO#(DataT) outbound1();
  mkSizedFIFO#(fifo_depth) the_outbound1(outbound1);
  FIFO#(DataT) outbound2();
  mkSizedFIFO#(fifo_depth) the_outbound2(outbound2);

  rule enq1 (True);
      DataT in_data = inbound1.first;
      FIFO#(DataT) out_queue  =
          determine_queue(in_data   ) == 0 ? outbound1 : outbound2;
      out_queue.enq(in_data   );
      inbound1.deq;
  endrule : enq1

endmodule  : ex_in1_out2_bs
```





www.bluespec.com

The Problem

I'm going to give you an informal (but hopefully clear) description of a part of a communication protocol, from Intel's new "QuickPath Interconnect".

(Involves concurrency, mutual recursion, "real time")

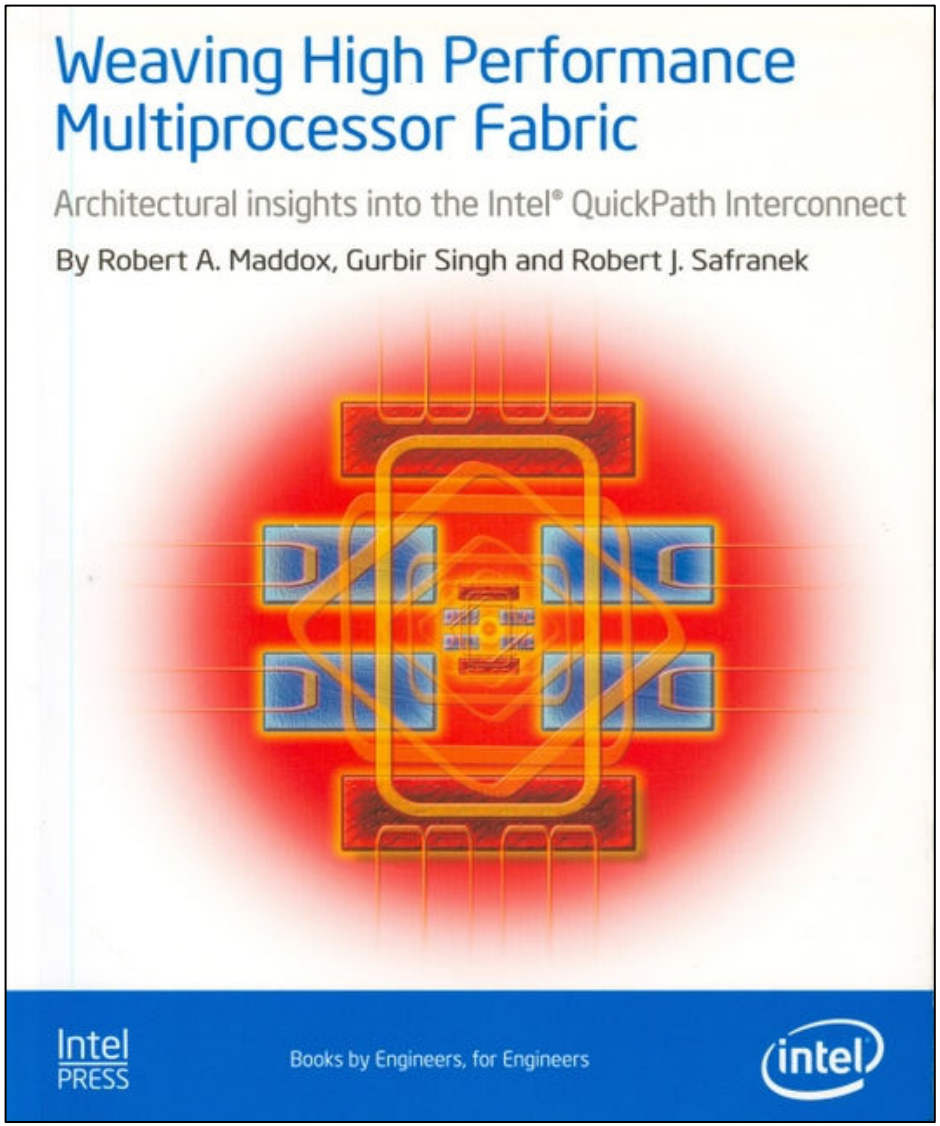
How would you formalize it in a programming language or specification language of your choice?

And, how would you prove correctness properties about it?

I can share a PDF of this problem description, to remind you of the details

If there is interest, I can show BSV code (Bluespec SystemVerilog) for this, later in the week (rewrite rules).
This code is synthesizable to hardware (via Verilog).

QPI information used in this presentation



Everything in this presentation about QPI is based only on information from the book pictured at left, which can be purchased from Intel Press and Amazon.com.

(no proprietary information)

bluespec

Where QPI is used

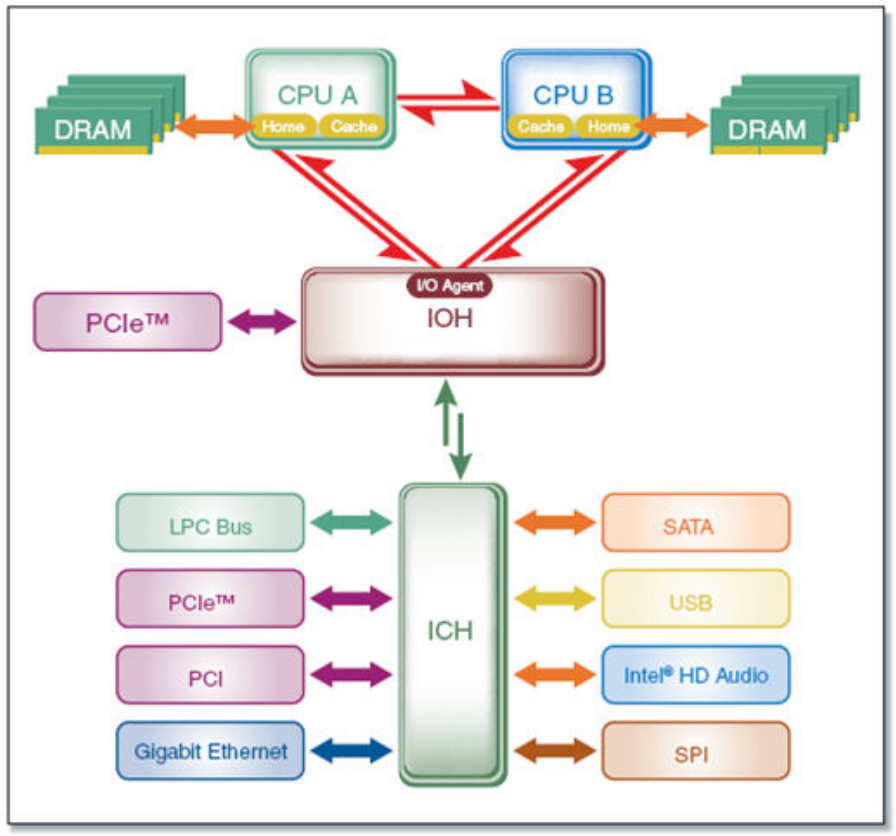


Figure 2.1 Two-Processor System Based on Intel® QuickPath Interconnect Technology

Intel's FSB (Frontside Bus) was used for the last 10 years to interconnect multiprocessor components. QPI replaces FSB, and is expected to be used for the next 10 years.

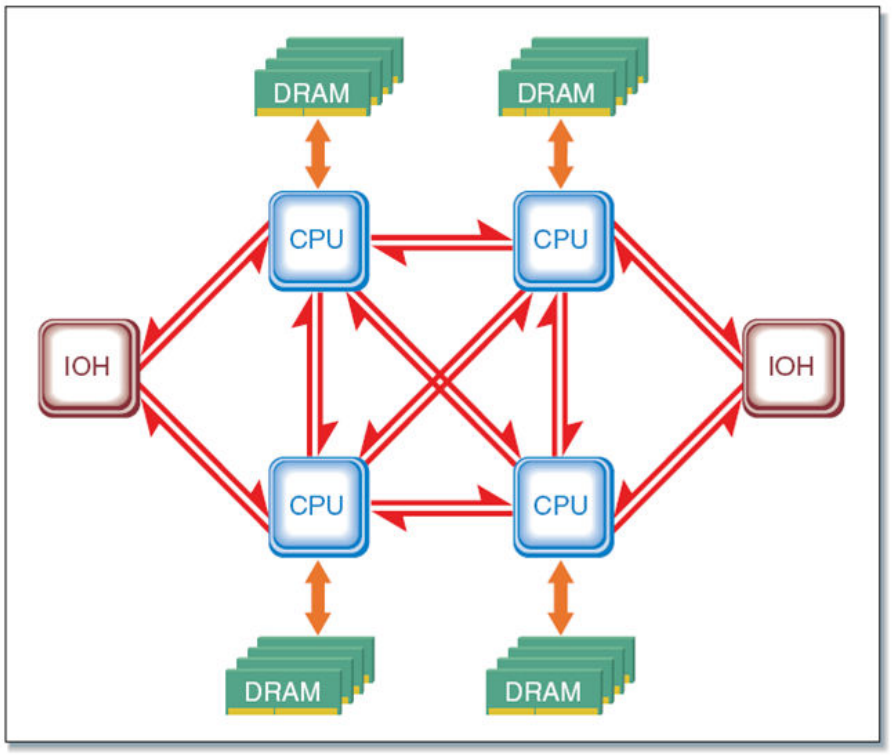


Figure 1.6 System with Four Processors Employing Intel® QuickPath Interconnect Technology

QPI link (point-to-point)

bluespec

QPI Protocol stack

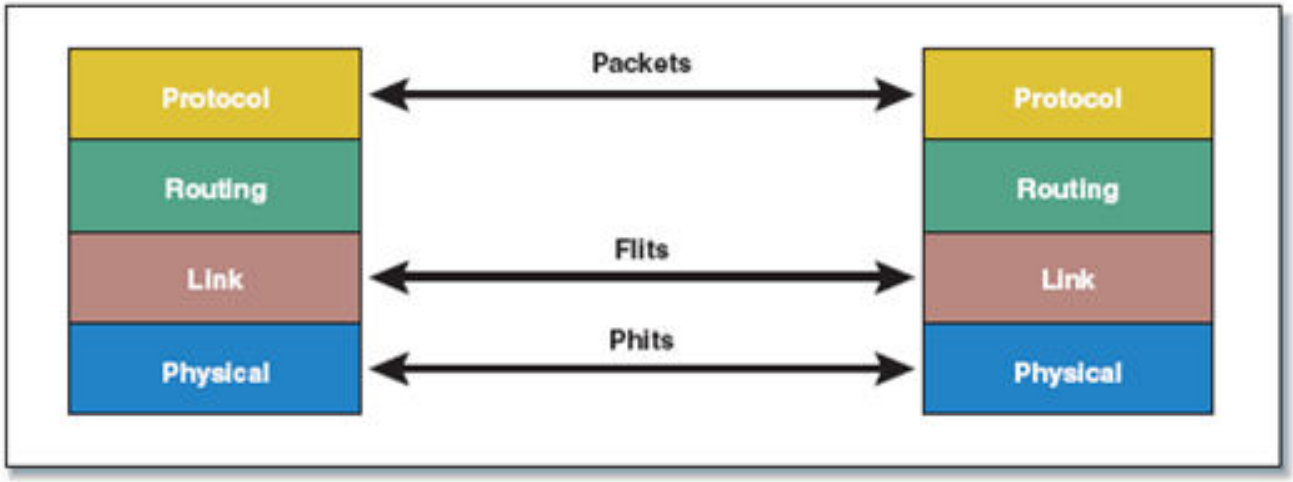


Figure 1.10 Layers Defined for the Intel® QuickPath Architecture

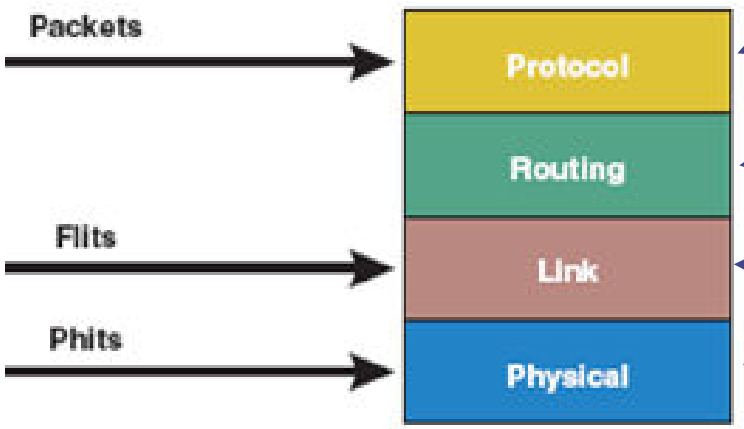
Packets: 1-13 Flits

Flits: 80 bits each

Phits: 20/40/80 bits each

bluespec

Context



Concerns communication of protocol packets for cache-coherence, uncached memory access, I/O, etc. Packets need 1 or more flits

Concerns forwarding for multi-hop communications

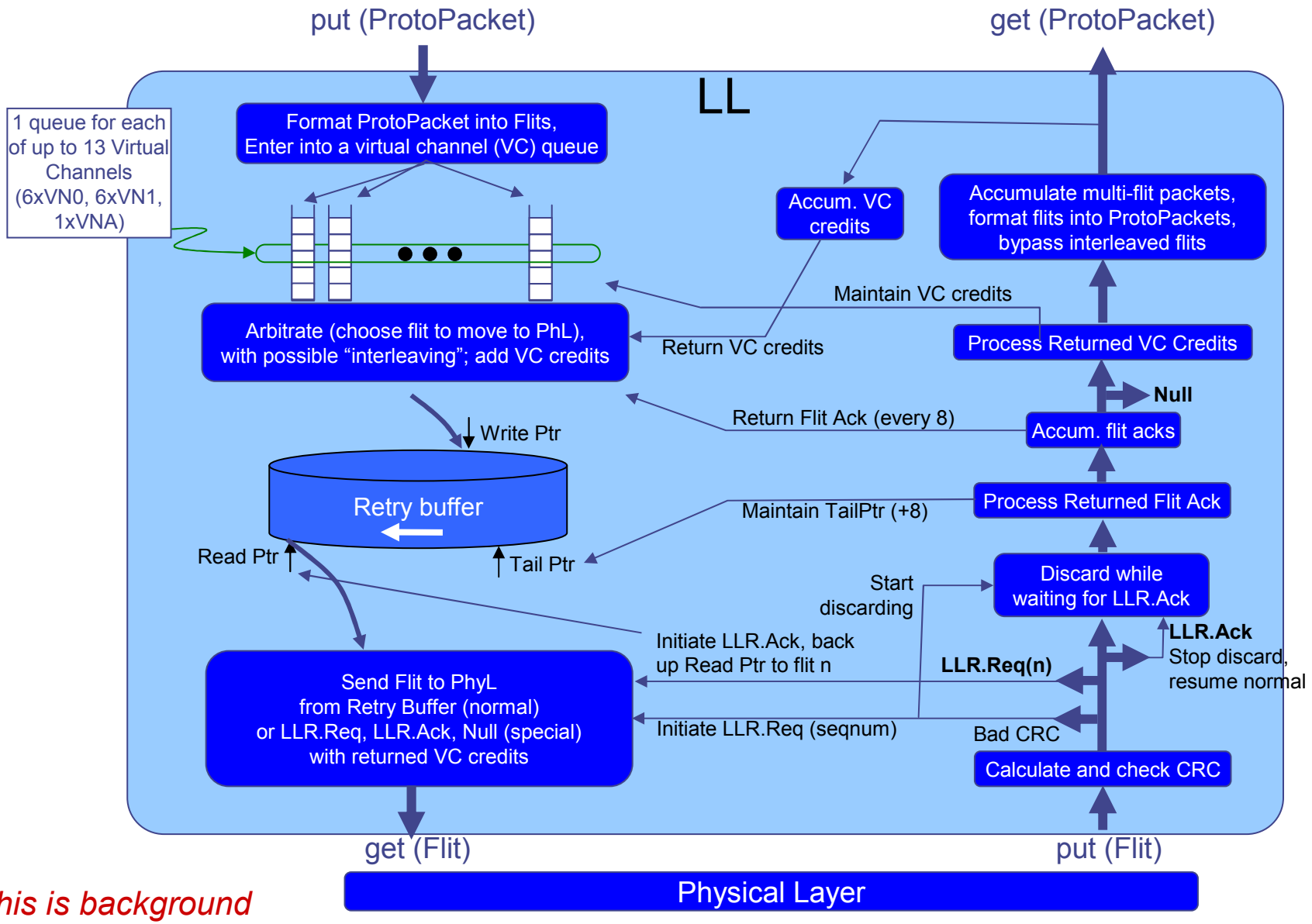
Concerns conversion of protocol packets into fixed-sized "flits", scheduling them over "virtual channels", and reliable delivery of flits on point-to-point links,

Focus of today's problem

Concerns reliable delivery of bits on point-to-point wires

bluespec

Link Layer (LL) detail (based on QPI book)

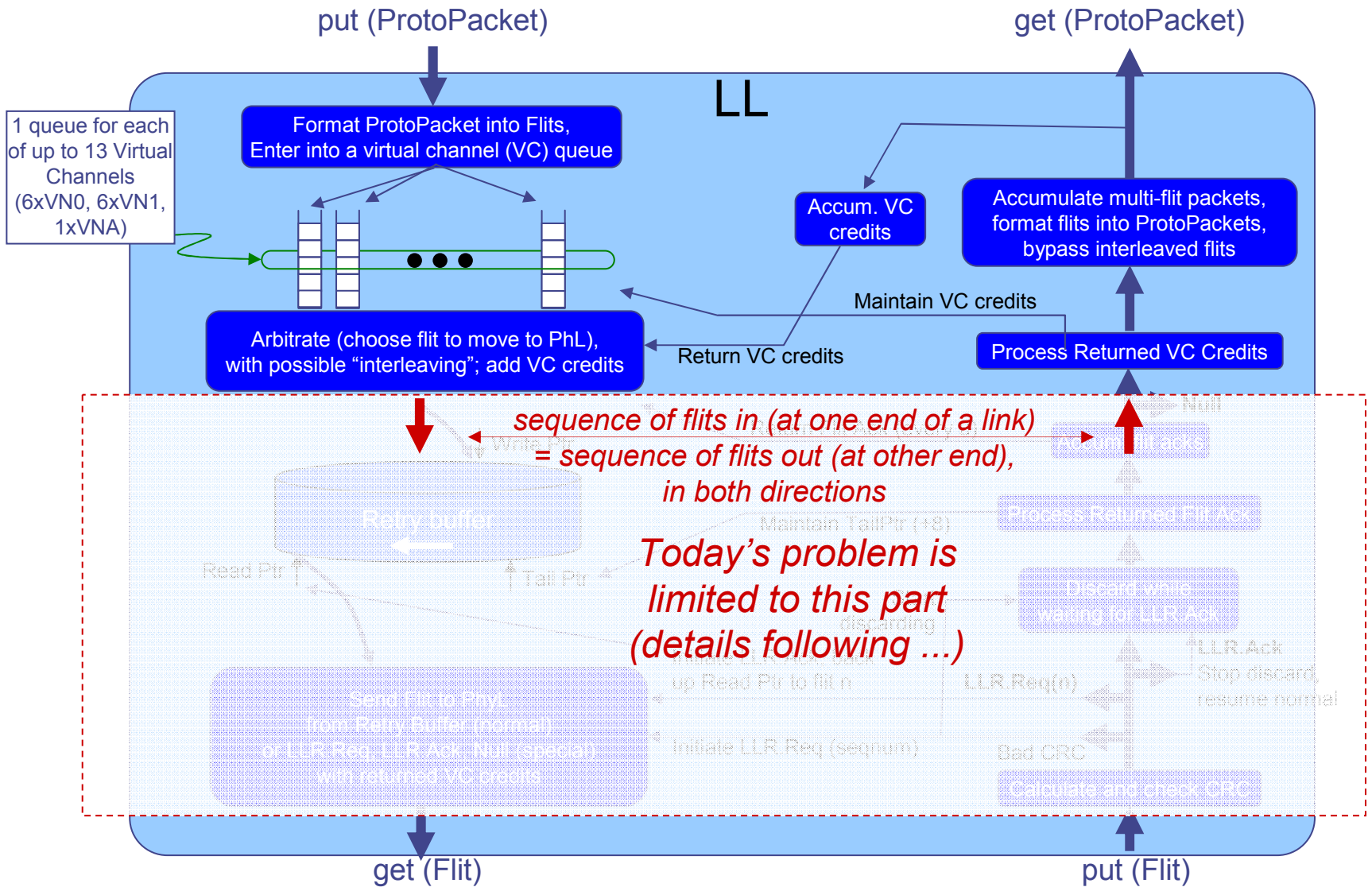


All this is background context; now begins today's problem description

Conceptually, every flit has a sequence number (starting at 0); not actually carried on flits

bluespec

Link Layer (LL) detail (based on QPI book)

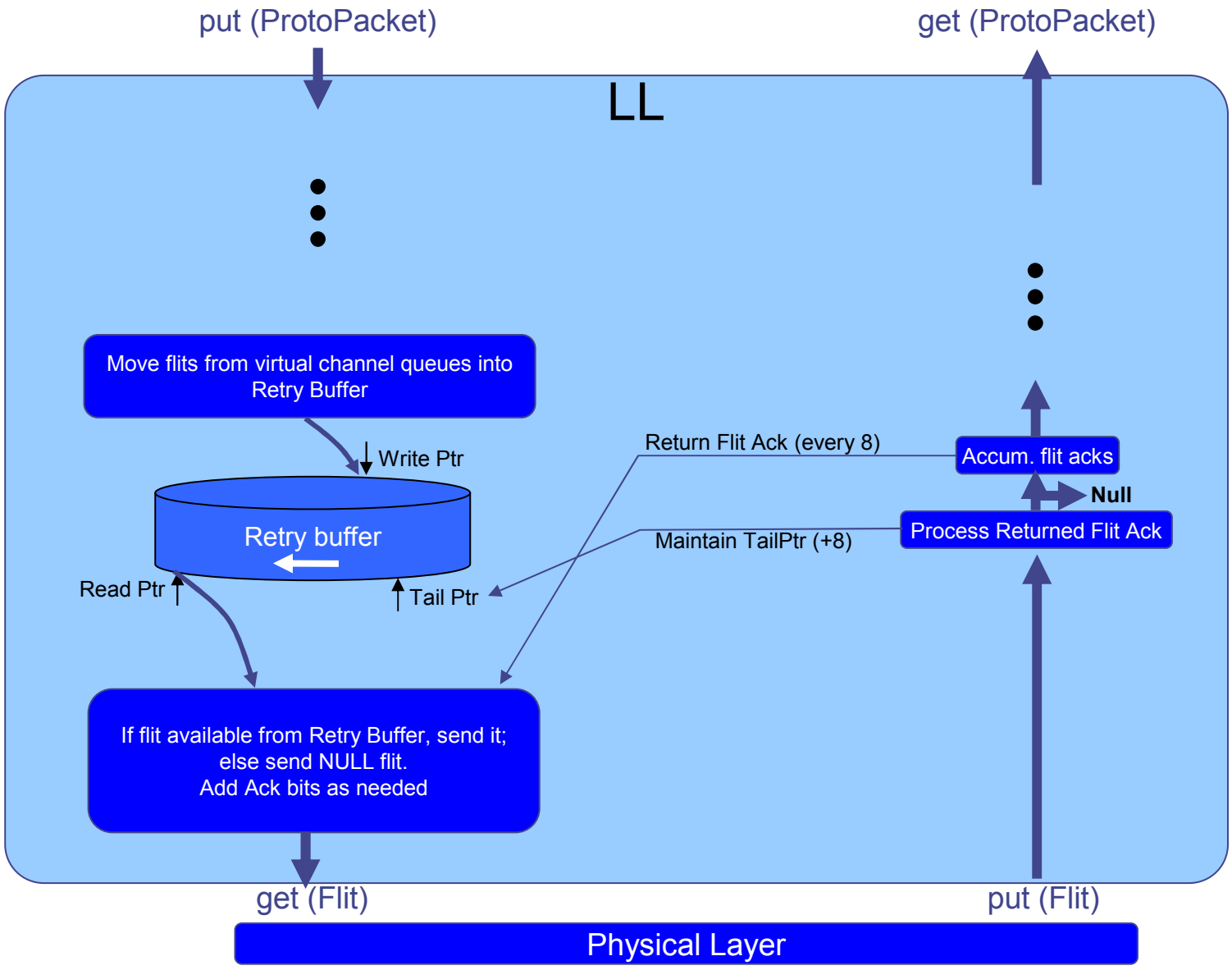


All this is background context; now begins today's problem description

Conceptually, every flit has a sequence number (starting at 0); not actually carried on flits

bluespec

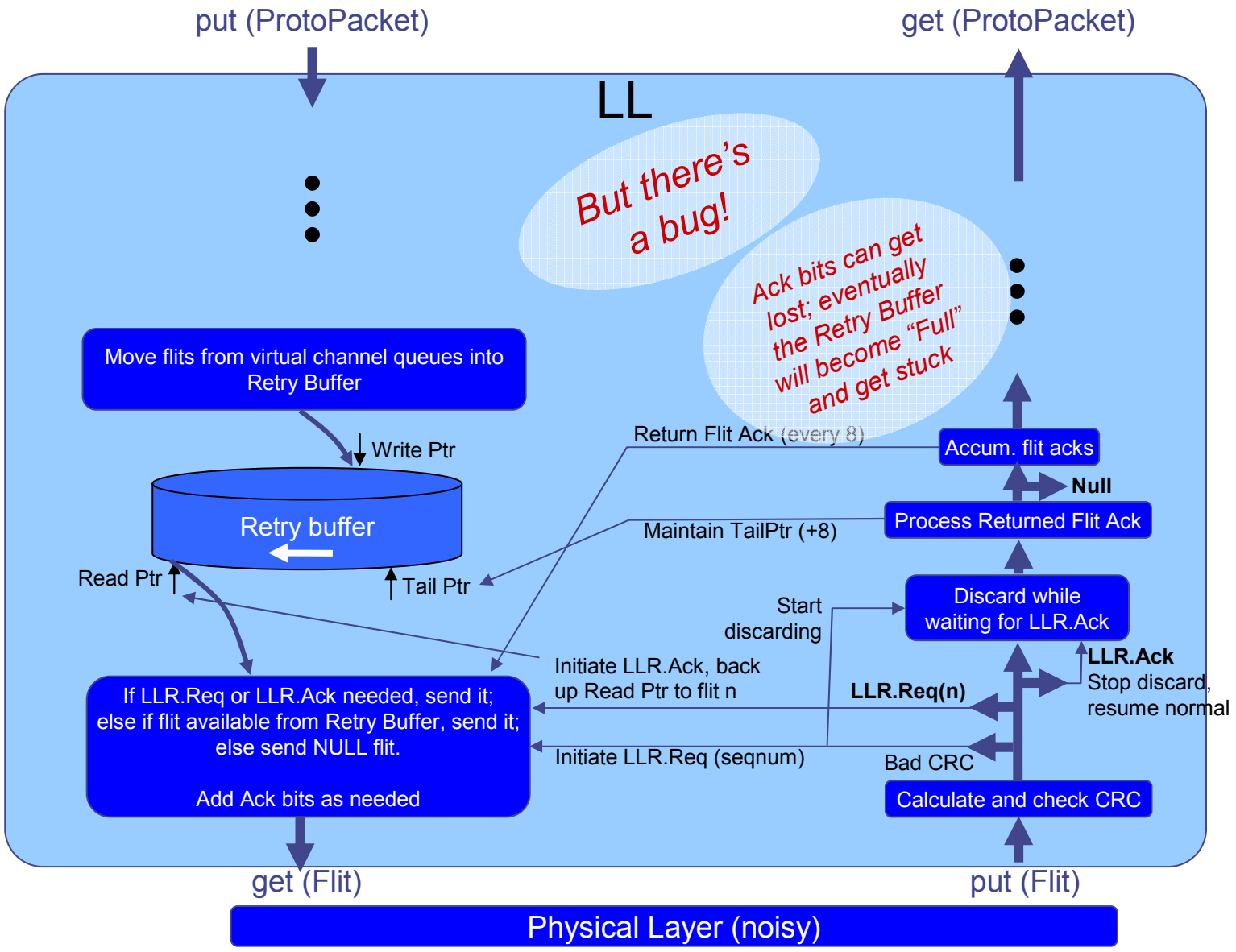
Moving flits: 1st approximation



A flit must be sent on every clock; if none available, send a NULL flit
For each 8 flits you receive, send an Ack bit back on a flit going the other way

bluespec

Moving flits: 2nd approximation

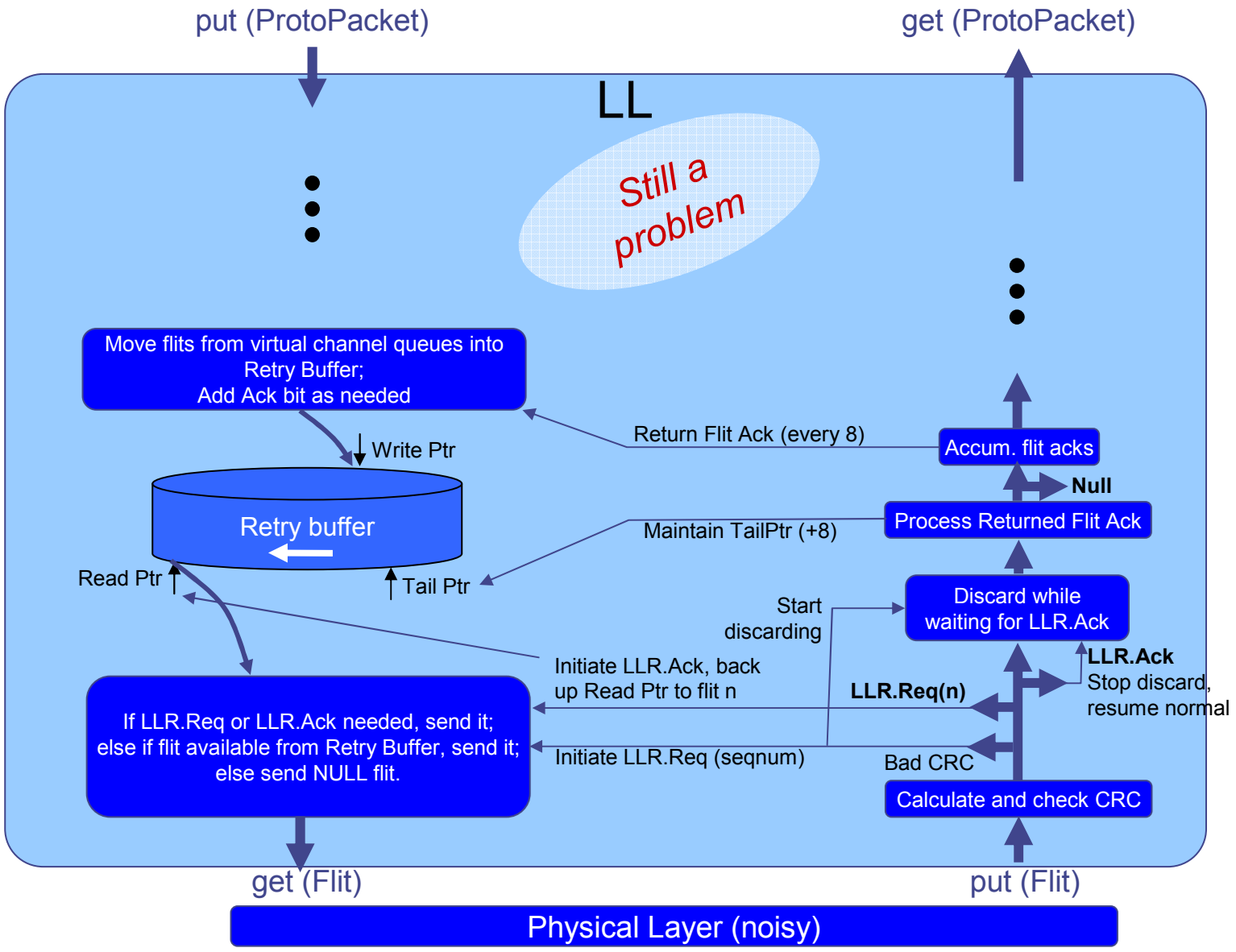


Flits may be corrupted in the Physical Layer; detect this using a CRC check

For now, assume LLR.Req and LLR.Ack flits are not themselves corrupted

bluespec

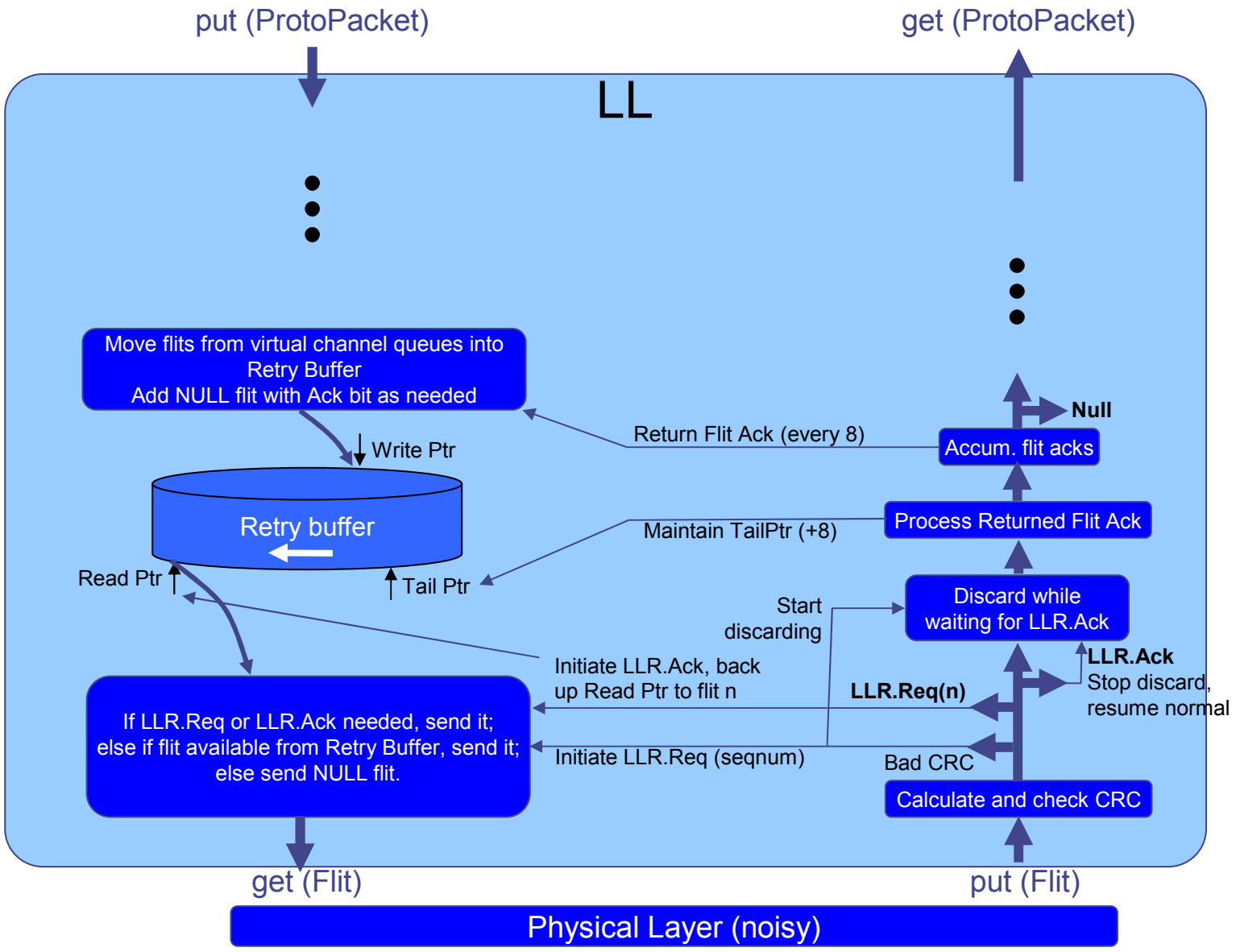
Moving flits: 2nd approximation (partially fixed)



Flits may be corrupted in the Physical Layer; detect this using a CRC check
For now, assume LLR.Req and LLR.Ack flits are not themselves corrupted

bluespec

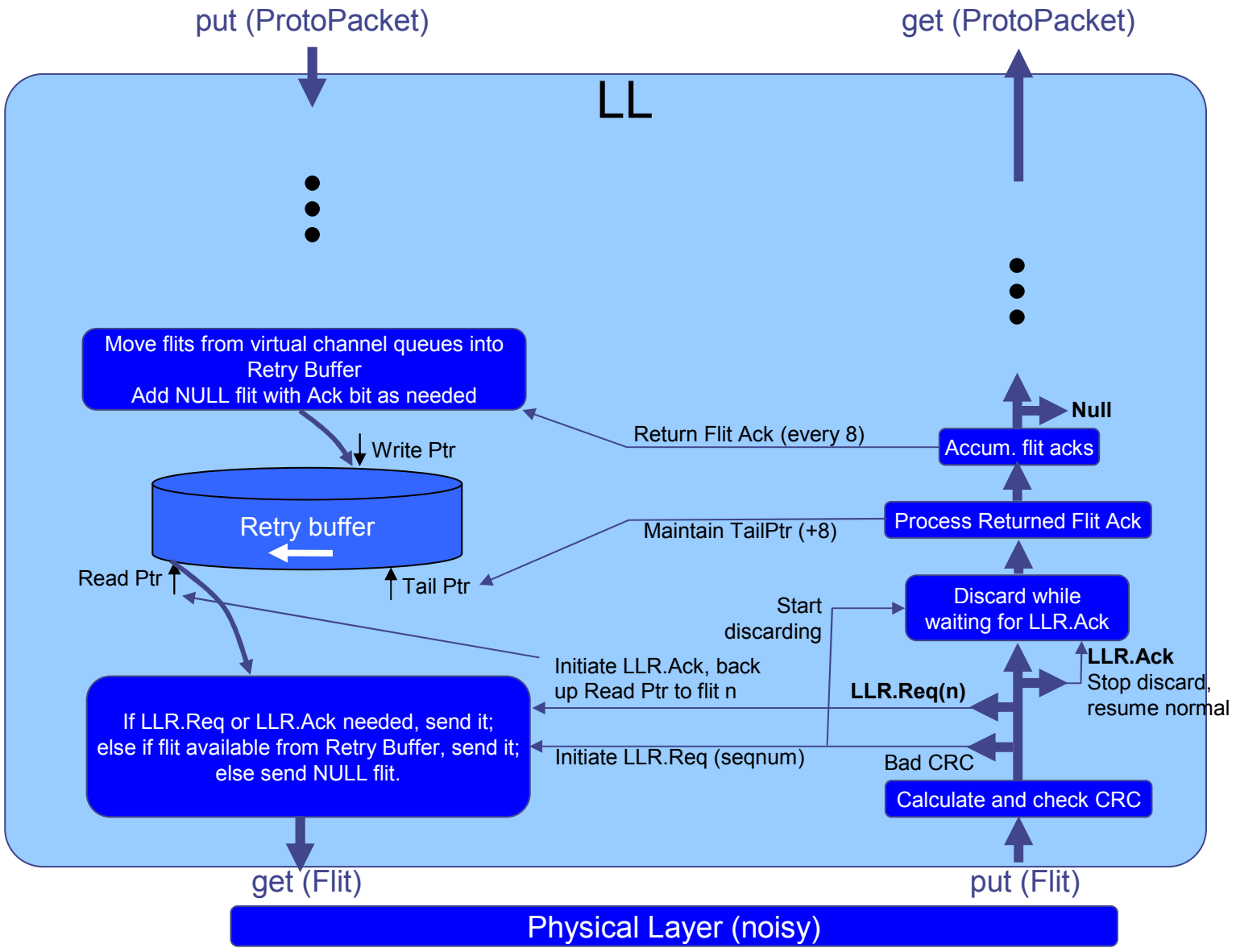
Moving flits: 2nd approximation (bug fixed)



Flits may be corrupted in the Physical Layer; detect this using a CRC check
For now, assume LLR.Req and LLR.Ack flits are not themselves corrupted

bluespec

Moving flits: final version (extra credit)



Now assume LLR.Req and LLR.Ack flits may also be corrupted (bad CRC).
Eventually, LL must go to "error state", but invent a mechanism that doesn't give up too easily

bluespec

bluespec™

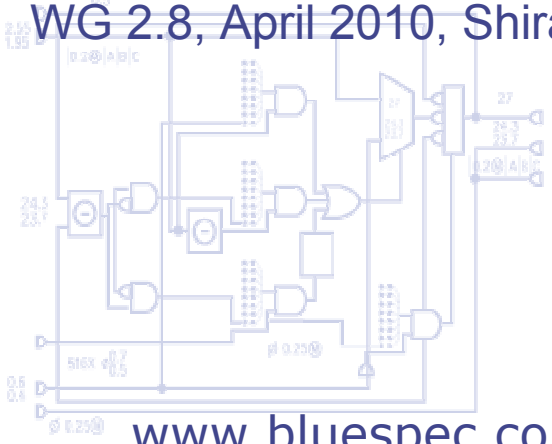
End

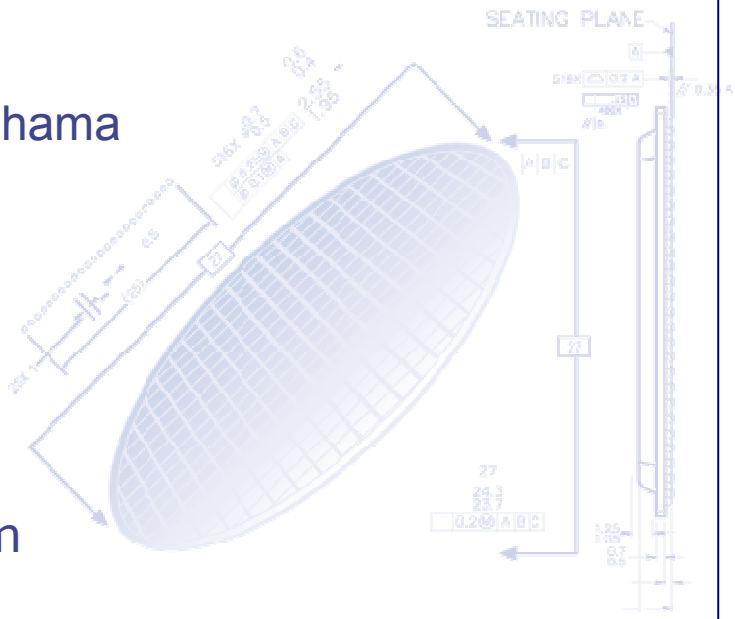
```
import FIFO::*;

typedef Bit#(32) DataT;

module ex_in1_out2_bs(Empty);

  Integer fifo_depth = 16;

  function Bit#(1)  determine_queue(DataT val );
    return (val[0]);
  endfunction

  FIFO#(DataT) inbound1();
  mkSizedFIFO#(fifo_depth) the_inbound1(inbound1);
  FIFO#(DataT) outbound1();
  mkSizedFIFO#(fifo_depth) the_outbound1(outbound1);
  FIFO#(DataT) outbound2();
  mkSizedFIFO#(fifo_depth) the_outbound2(outbound2);

  rule enq1 (True);
      DataT in_data = inbound1.first;
      FIFO#(DataT) out_queue  =
          determine_queue(in_data   ) == 0 ? outbound1 : outbound2;
      out_queue.enq(in_data   );
      inbound1.deq;
  endrule : enq1

endmodule  : ex_in1_out2_bs
```

Slides accompanying a code-walkthrough of the BSV solution to the QPI problem

Rishiyur S. Nikhil

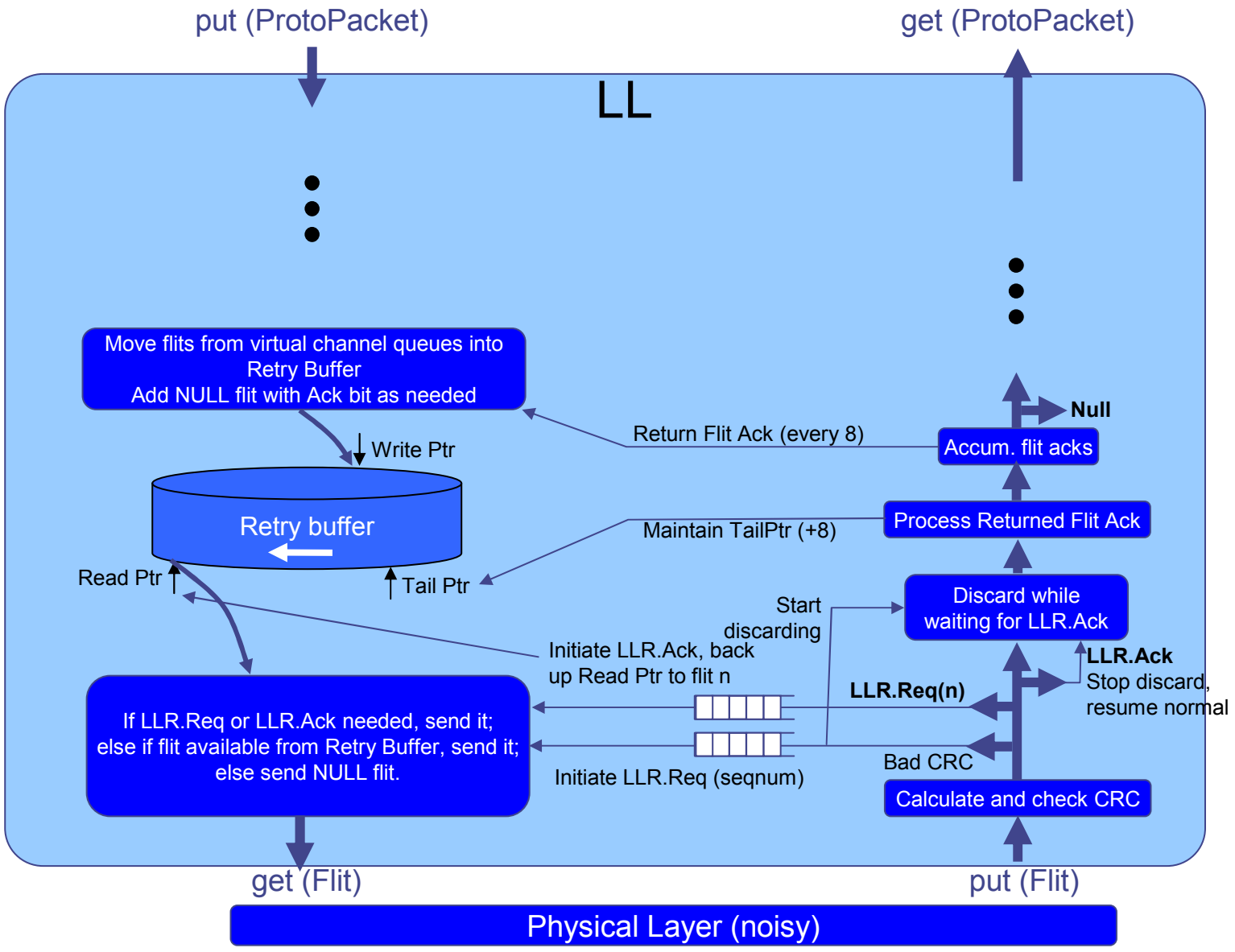
WG 2.8, April 2010, Shirahama



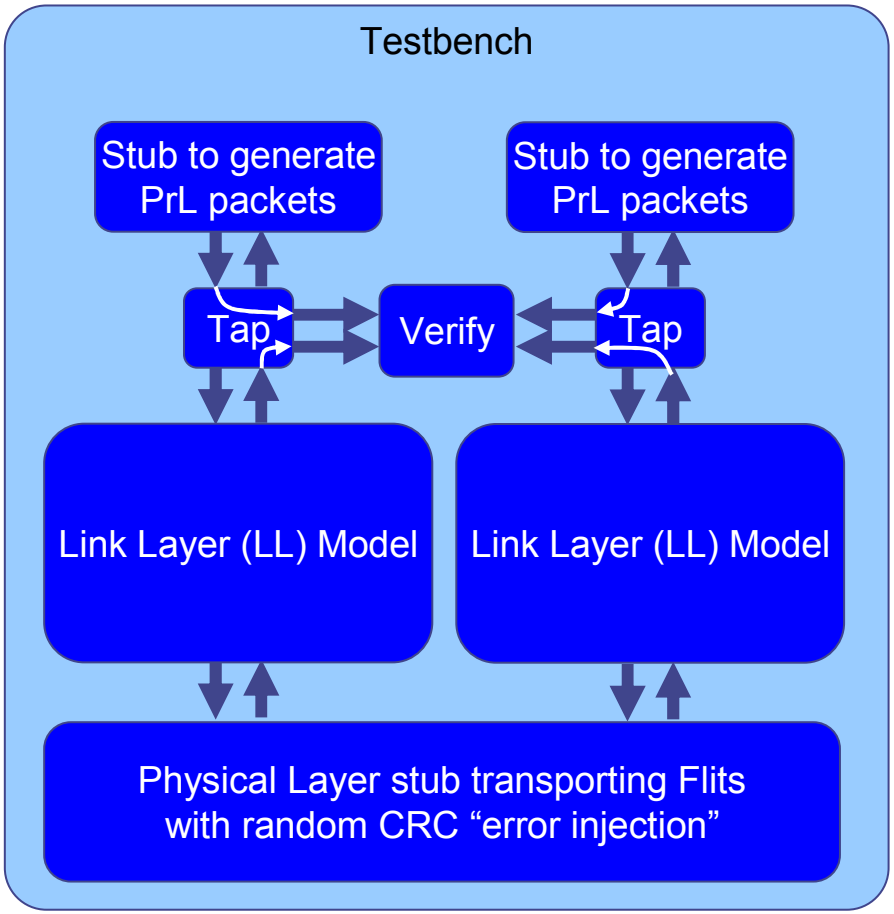


www.bluespec.com

Moving flits



Module structure



bluespec