GT

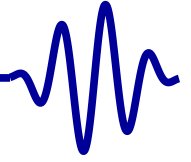## Gisselquist Technology, LLC

An AXI Network Protocol

Daniel E. Gisselquist, Ph.D.
March, 2023

AXIN Protocol

Motivation: ABORT

1. AXI stream requires backpressure support
2. Few devices can truly handle backpressure
3. Typical solution:

   - Store a packet in memory (Block RAM)
   - Abort packets that run out of memory before completion
   - Once a packet is fully stored, feed it out via AXI stream

4. Typical solution limits the maximum packet size

Solution: Add a new field to allow the source to ABORT a packet being streamed.

Motivation: BYTES

1. High speed interfaces require large beat sizes
2. Beat size increases when clock domain crossings

   □ Ex: GbE requires one octet per beat initially, 2-4 octets per beat if crossing clock domains
   □ Ex: 10GbE requires 4 octets per beat, 8-16 if crossing clock domains

3. Packet length requires octet level precision

Motivation: BYTES

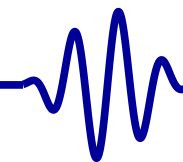1. AXI Stream solution

   □ Uses TSTRB and TKEEP
   □ Allows "null" bytes, empty beats, and position bytes
   □ None of these make sense in a packet context
   □ Further, they use more data than necessary

2. A packed stream only needs to know the number of valid bytes on the last beat.

Proposal: Replace TSTRB and TKEEP with a BYTES field

□ BYTES indicates the number of bytes valid in any given beat
□ Only less than full on the last beat.
□ Uses $clog2(DW/8) bits, rather than 2*(DW/8).

   – Can reduce logic requirements by up to 18%

Signals

Global Signals	AXI Stream	AXIN
ACLK ARESETN	TVALID TREADY	VALID READY
	TDATA[DW-1:0] TSTRB[DW/8-1:0] TKEEP[DW/8-1:0] TLAST TUSER	DATA[DW-1:0] BYTES[BW-1:0] LAST ABORT

Where DW is the bits per beat, and BW=$clog2(DW/8)

Signals

Global Signals	AXI Stream	AXIN
ACLK ARESETN	TVALID TREADY	VALID READY
	TDATA[DW-1:0] ~~TSTRB[DW/8-1:0]~~ ~~TKEEP[DW/8-1:0]~~ TLAST ~~TUSER~~	DATA[DW-1:0] BYTES[BW-1:0] LAST ABORT

These signals arean't really needed

Where `DW` is the bits per beat, and `BW=$clog2(DW/8)`

Signals

Global Signals	AXI Stream	AXIN
ACLK ARESETN	TVALID TREADY	VALID READY
	TDATA[DW-1:0] ~~TSTRB[DW/8-1:0]~~ ~~TKEEP[DW/8-1:0]~~ TLAST ~~TUSER~~	DATA[DW-1:0] BYTES[BW-1:0] LAST ABORT

We'll add these two new ones

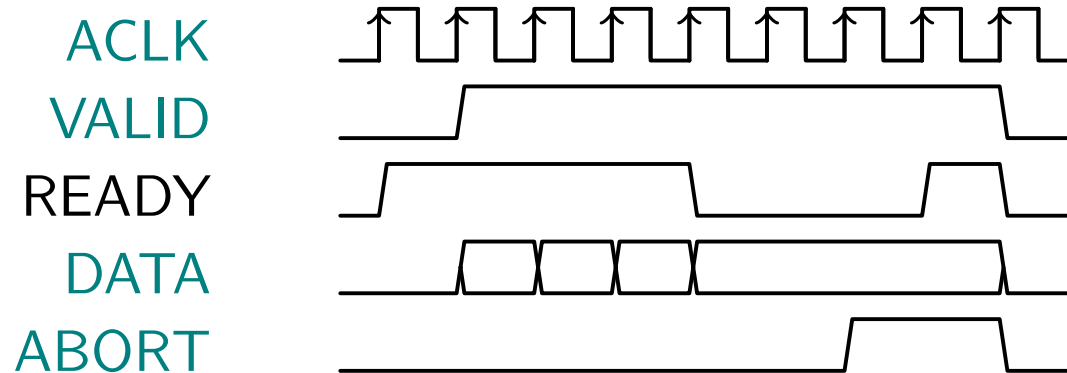Where `DW` is the bits per beat, and `BW=$clog2(DW/8)`

ABORT Rules

1. ABORT may be raised at any time

It may be raised while `VALID && !READY`.

ABORT Rules
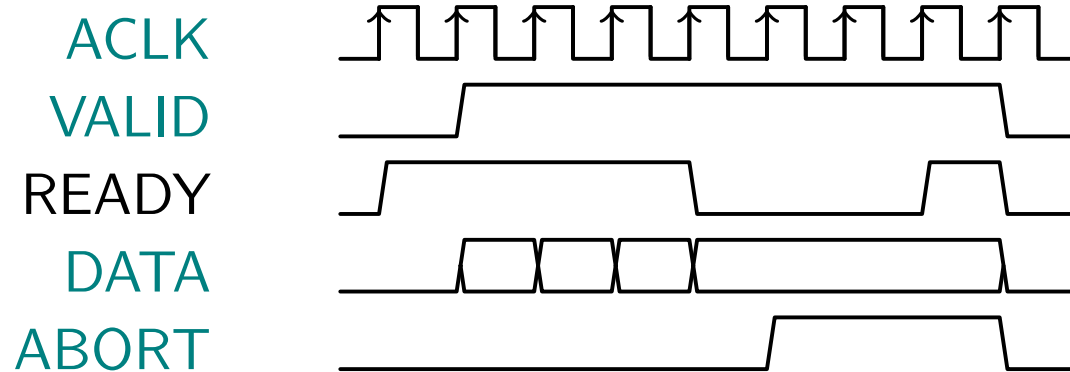
1. ABORT may be raised at any time
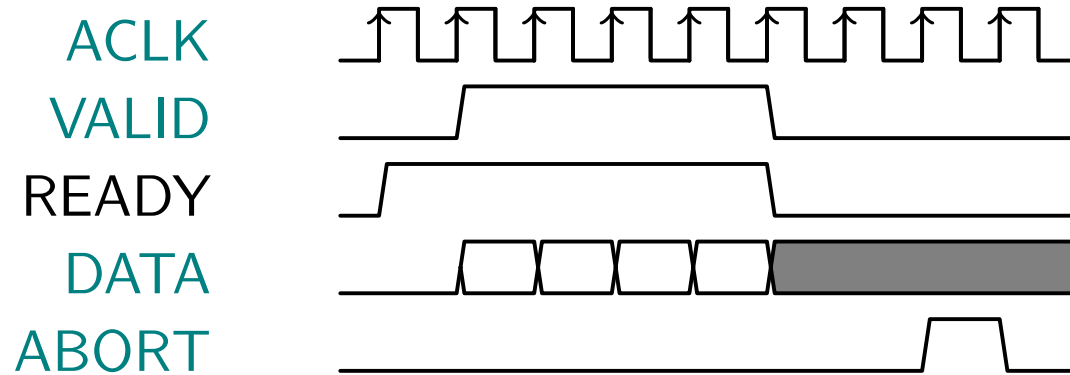
It may be raised even without raising `VALID`.

ABORT Rules

1. ABORT may be raised at any time
2. ABORT may be only be released if not stalled


```
property assert (@(posedge ACLK)
        disable iff (!ARESETN)
        (VALID && !READY && ABORT)
                |=> VALID && ABORT);
```

ABORT Rules

1. ABORT may be raised at any time
2. ABORT may be only be released if not stalled
3. While legal, it doesn't make sense to raise ABORT if `VALID && LAST`.

4.
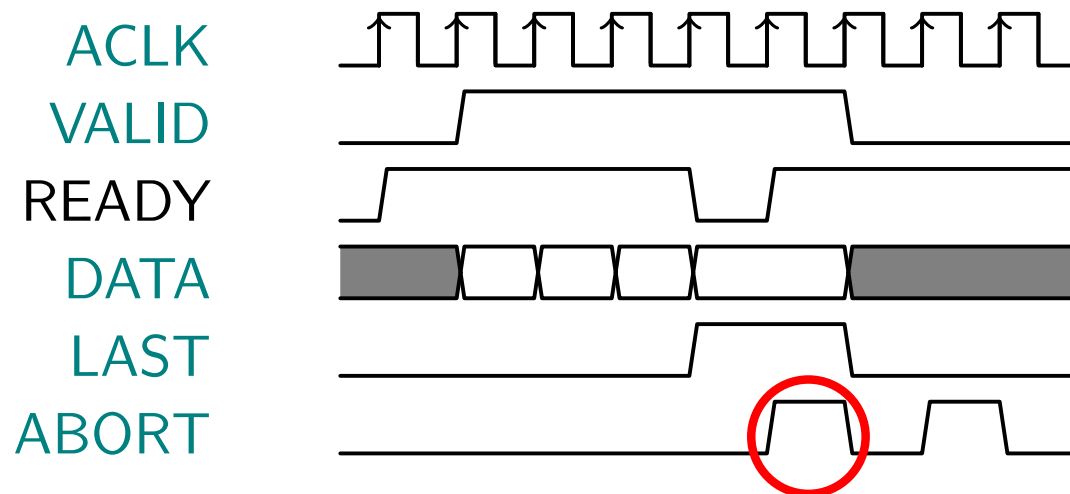
If `VALID && LAST` are both true, the packet has successfully, entered the downstream registers. Raising ABORT at this point makes little sense.

ABORT Rules

1. ABORT may be raised at any time
2. ABORT may be only be released if not stalled
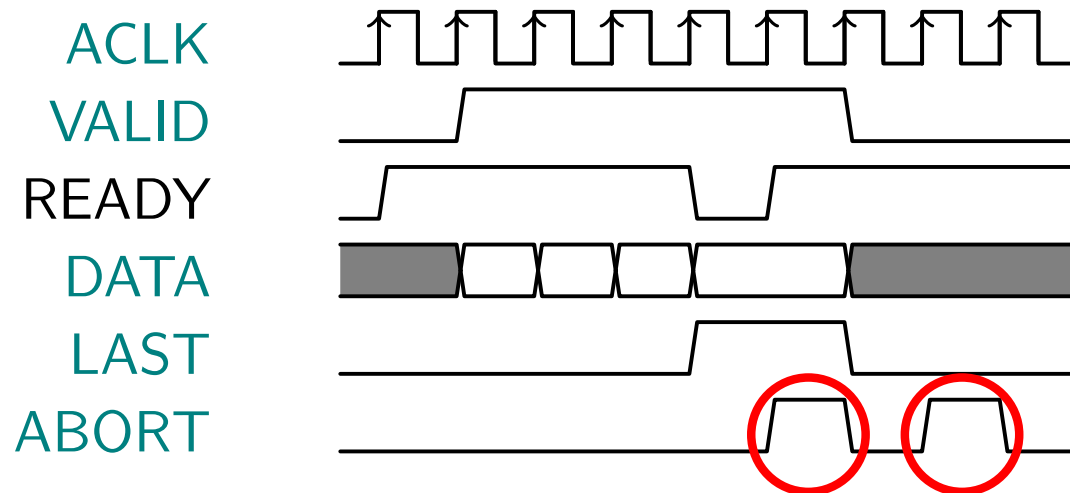3. While legal, it doesn't make sense to raise ABORT if `VALID && LAST`.
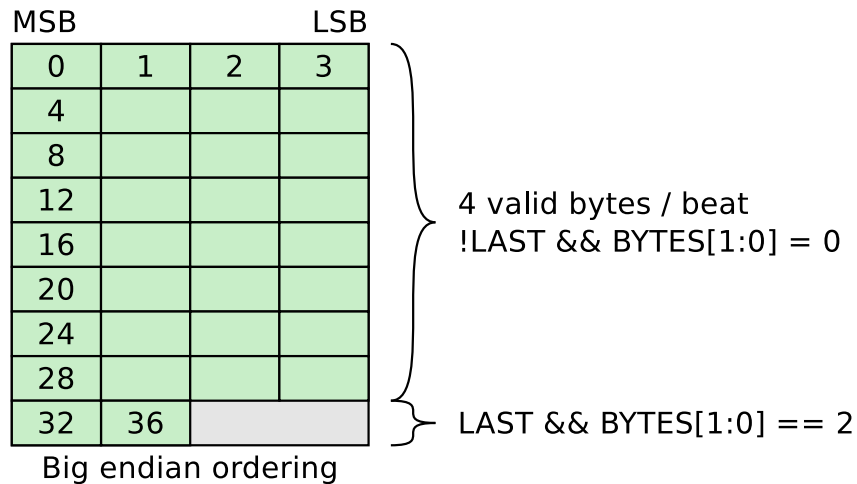4. Nor does it make sense to raise ABORT if a packet hasn't started.

BYTES Rules

1. All beats are packed
2. The BYTES field contains $clog2(DW/8)$ bits.
   If ever BYTES $== 0$, then there are DW/8 valid bytes implied.
3. Only the last beat can contain fewer than DW/8 bytes
4. The last beat cannot be empty. It must have at least one byte.
5. Byte ordering can be big   endian

0	1	2	3
4			
8			
12			
16			
20			
24			
28			
32	36		

MSB ... LSB

4 valid bytes / beat
!LAST && BYTES[1:0] = 0

LAST && BYTES[1:0] == 2

Big endian ordering

BYTES Rules

1. All beats are packed
2. The BYTES field contains $clog2(DW/8) bits.
   If ever BYTES $== 0$, then there are DW/8 valid bytes implied.
3. Only the last beat can contain fewer than DW/8 bytes
4. The last beat cannot be empty. It must have at least one byte.
5. Byte ordering can be big or little endian

MSB   LSB

3	2	1	0
			4
			8
			12
			16
			20
			24
			28
		36	32

4 valid bytes / beat
!LAST && BYTES[1:0] = 0

LAST && BYTES[1:0] == 2

Little endian ordering