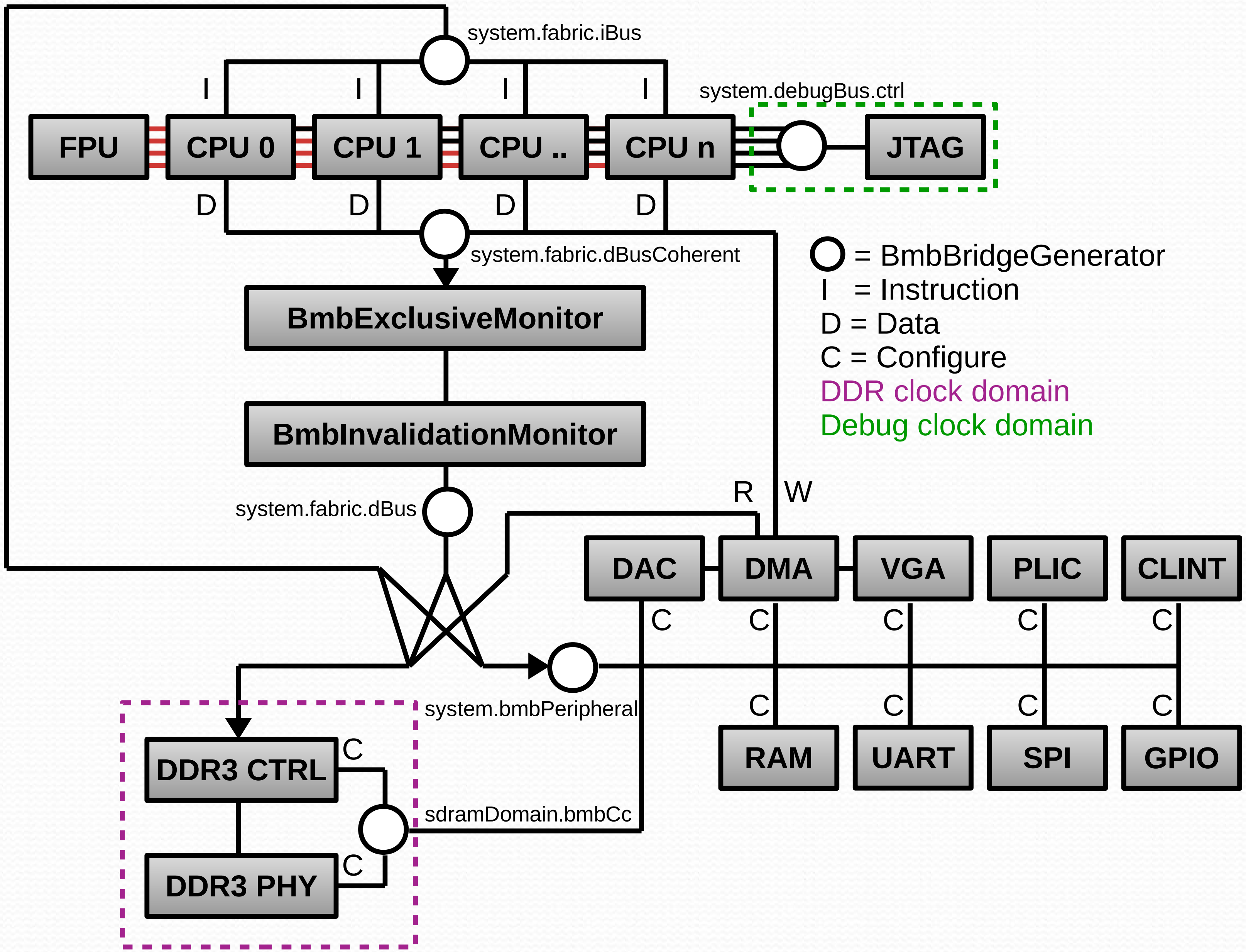
SpinalHDL

SaxonSoc

In short

- Started in parallel to VexRiscv SMP
- Based on
  - SpinalHDL “Handle” tool
  - Banana Memory Bus
- Key features
  - Memory atomics
  - Memory coherency
  - SDR/DDR2/DDR3 controller
  - Multi-core Linux
  - BMB (Memory bus)

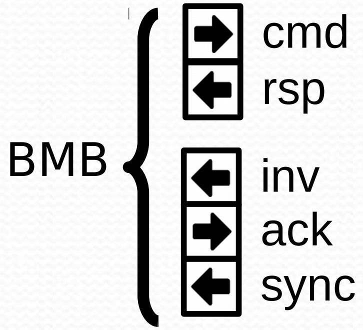
ArtyA7 port



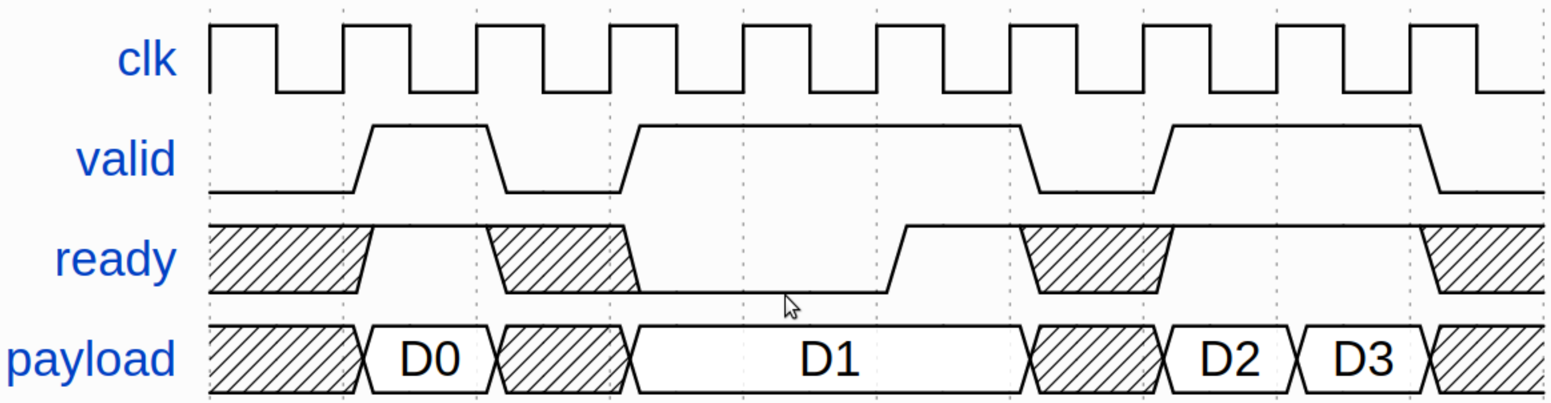
○ = BmbBridgeGenerator
I = Instruction
D = Data
C = Configure
DDR clock domain
Debug clock domain

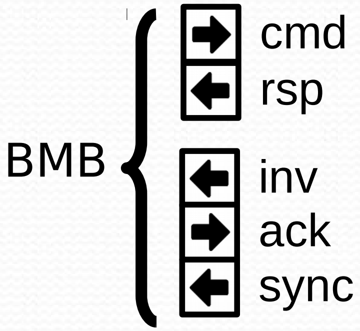
BMB (Banana Memory Bus)

- Goal :
  - Allow both cache-less and cache-full designs (latency / bandwidth)
  - Avoid FPGA expensive interconnects
  - No license of any sort
- Key concepts
  - cmd / rsp streams for accesses
  - cmd.context retrieved in rsp.context
  - [inv / ack / sync] stream for cache invalidation
  - [out of order]
  - Can be read write / read only / write only
  - Atomic support via load reserve / store conditional



BMB stream




BMB

cmd

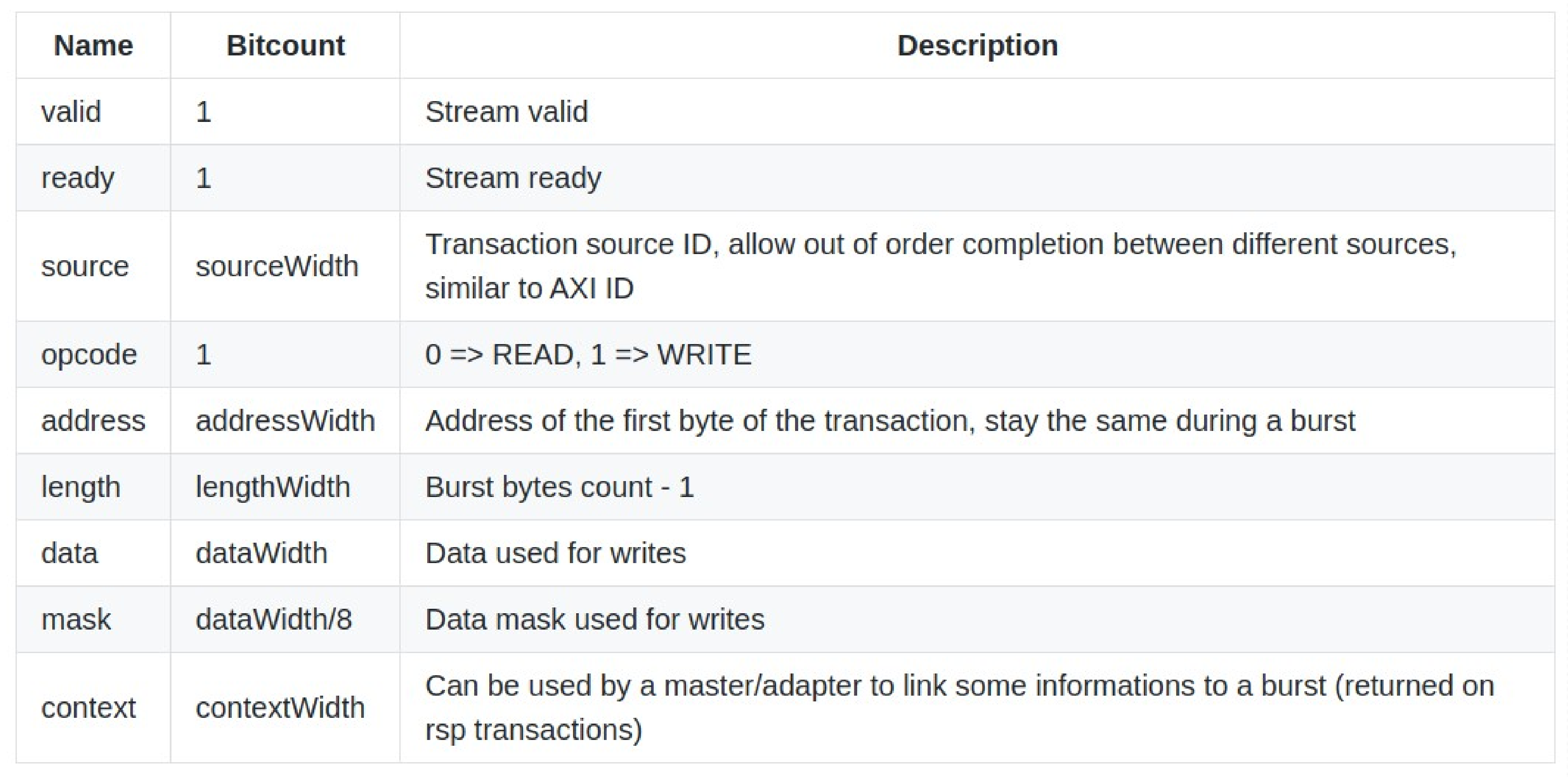
rsp

inv

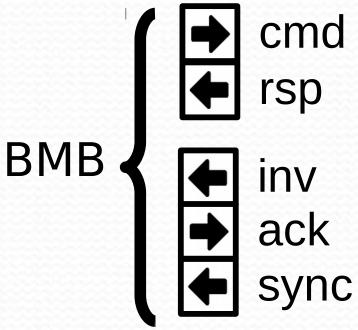
ack

sync

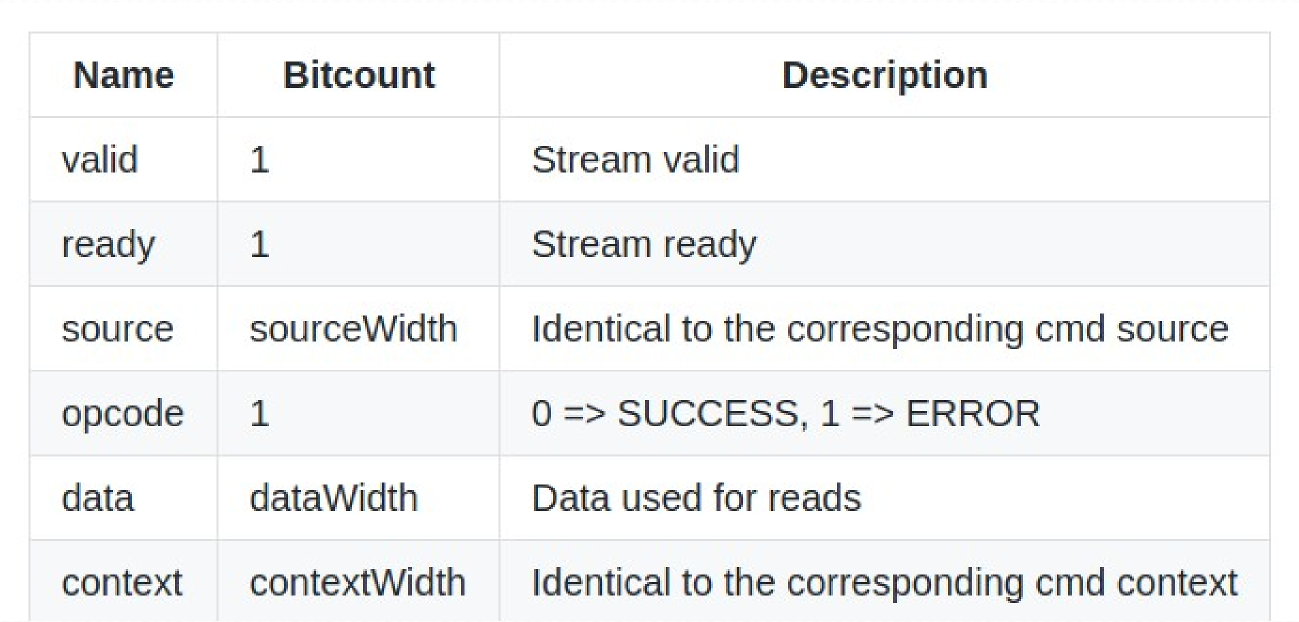
cmd signals



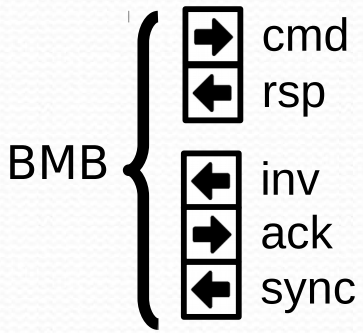
Name	Bitcount	Description
valid	1	Stream valid
ready	1	Stream ready
source	sourceWidth	Transaction source ID, allow out of order completion between different sources, similar to AXI ID
opcode	1	0 => READ, 1 => WRITE
address	addressWidth	Address of the first byte of the transaction, stay the same during a burst
length	lengthWidth	Burst bytes count - 1
data	dataWidth	Data used for writes
mask	dataWidth/8	Data mask used for writes
context	contextWidth	Can be used by a master/adapter to link some informations to a burst (returned on rsp transactions)



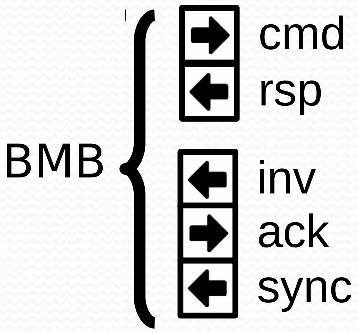
rsp signals

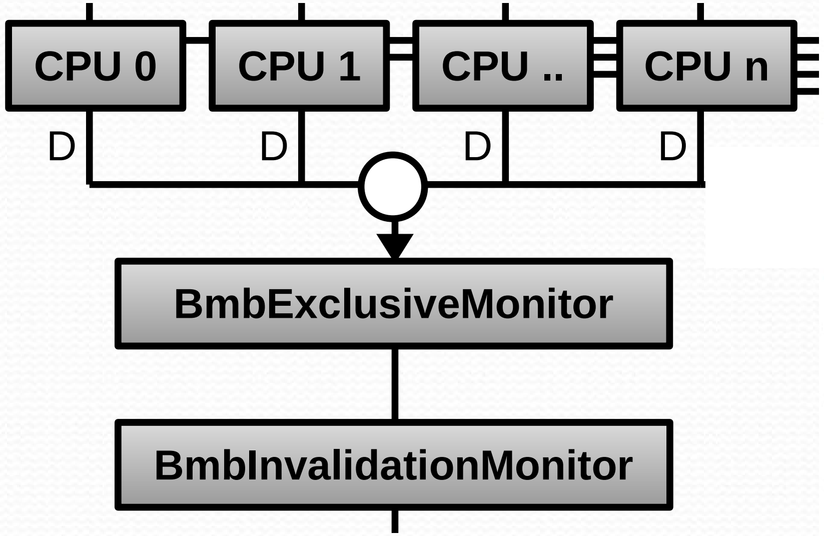


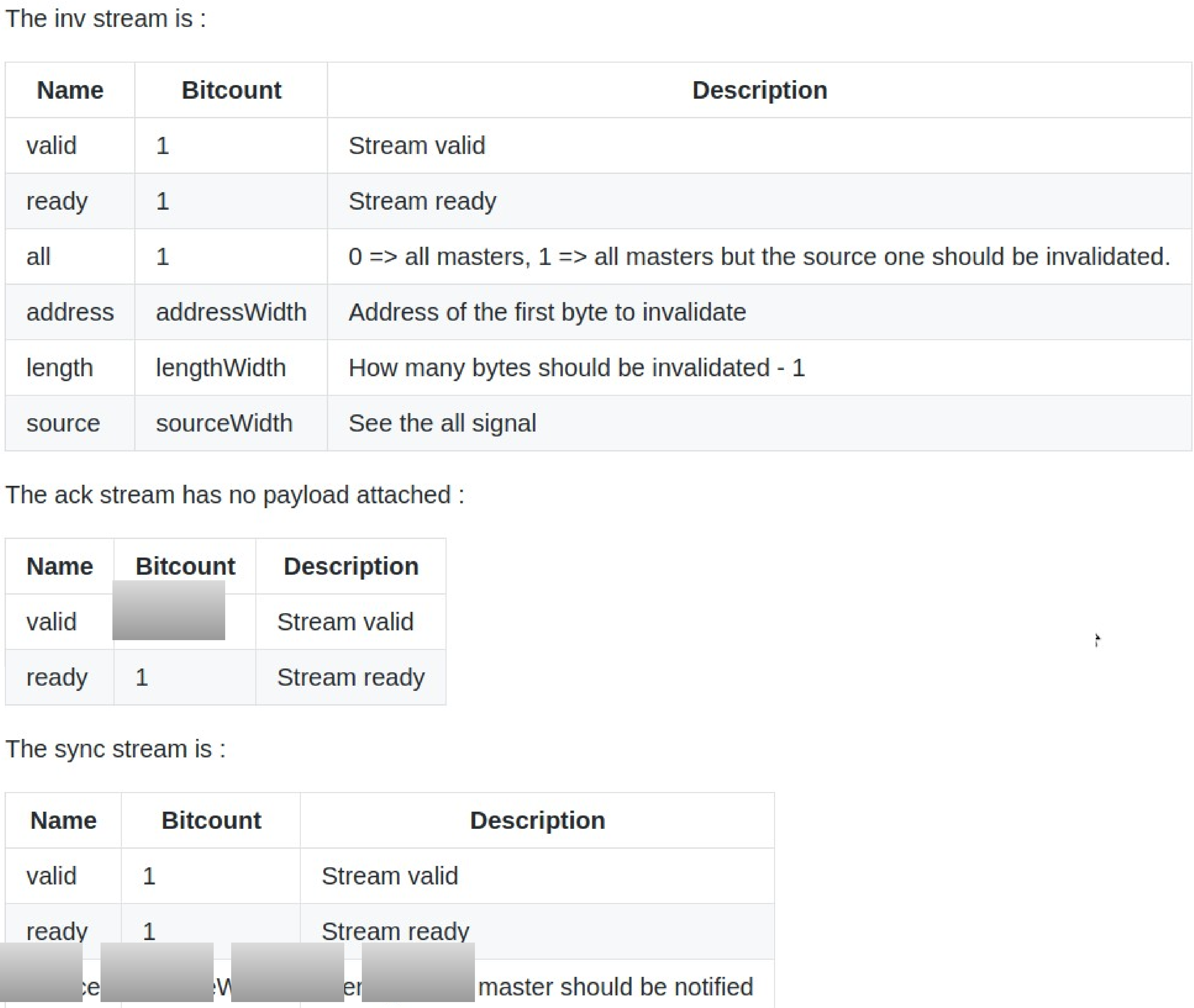
Name	Bitcount	Description
valid	1	Stream valid
ready	1	Stream ready
source	sourceWidth	Identical to the corresponding cmd source
opcode	1	0 => SUCCESS, 1 => ERROR
data	dataWidth	Data used for reads
context	contextWidth	Identical to the corresponding cmd context



inv/ack/sync







The inv stream is :

Name	Bitcount	Description
valid	1	Stream valid
ready	1	Stream ready
all	1	0 => all masters, 1 => all masters but the source one should be invalidated.
address	addressWidth	Address of the first byte to invalidate
length	lengthWidth	How many bytes should be invalidated - 1
source	sourceWidth	See the all signal

The ack stream has no payload attached :

Name	Bitcount	Description
valid	[redacted]	Stream valid
ready	1	Stream ready

The sync stream is :

Name	Bitcount	Description
valid	1	Stream valid
ready	1	Stream ready
[redacted]ce	[redacted]eW[redacted]	[redacted]er[redacted] master should be notified

Handle API

- Allow :
  - Solve elaboration chunks concurrently
  - “Lazy” specification
  - Define how something should be done before having all the information you need

Handle API dummy example

```
// Create two empty Handles
val a, b = Handle[Int]

// Create a Handle which will be loaded asynchronously by the given body result
val calculator = Handle {
    a.get + b.get // .get will block until they are loaded
}

// Same as above
val printer = Handle {
    println(s"a + b = ${calculator.get}") // .get is blocking until the calculator body is done
}

// Synchronously load a and b, this will unblock a.get and b.get
a.load(3)
b.load(4)
```