Wishbone: a free SoC bus family

Tristan Gingold – FSiC 2022

What is Wishbone ?

- A SoC Bus
  - Like AXI4, AXI3, AHB, APB, Avalon, CoreConnect
  - Teaser: More APB/AXI4-Lite than AHB/AXI4
- Initial specifications by Wade D. Peterson
- Public domain specification
  - OpenSource friendly
- You can use the Wishbone logo if you follow the specs

Why Wishbone ?

- Free license, free of charge
- Widely used in FOSS projects
- Many cores available
  - I2c, spi, cpus, timers, bridges, eth, crypto, ...
  - Main SoC bus for opencores.org
- Use in industry ?
- Simple
- Quite good specification
- Many examples

History

- Revision A (preliminary) - June 16, 1999
- Revision A.1 (preliminary) - Updated July 27, 1999
- Revision B (preliminary) – Updated January 5, 2001
- Revision B.1 (preliminary) – Updated January 8, 2001
  - Remove all copyright notices and place into the public domain.
- Revision B.2, Released: October 10, 2001
- Revision B.3, Released: September 7, 2002
  - Richard Herveille, OpenCores Organization
- Revision B.4 - 2010

The Document

- B3: Notice is hereby given that this document is not copyrighted, and has been placed into the public domain. It may be freely copied and distributed by any means
- B4: Copyright Notice
  - This ebook is Copyright © 2010 OpenCores
  - (and with the above text from B3)

- Document sources were not available
- Recreated in markdown:
  - https://github.com/fossi-foundation/wishbone
  - From B3

A Family

- Any address size
- 8, 16, 32 or 64 bit data size
- 8, 16, 32 or 64 bit data granularity
- Any endianness

Like many/most SoC bus, but:

- Optional signals (TAG, RTY, ERR, LOCK)
- 3 flavours: classic, pipelined or registered

Main principles

- Master/Slave
- Clock synchronous
- Unidirectional wires (data in and data out buses)
- Address decoding by master
  - A slave is targeted on every transaction
  - A slave has to reply (no timeout)
- Strobe / Ack mechanism

Naming convention

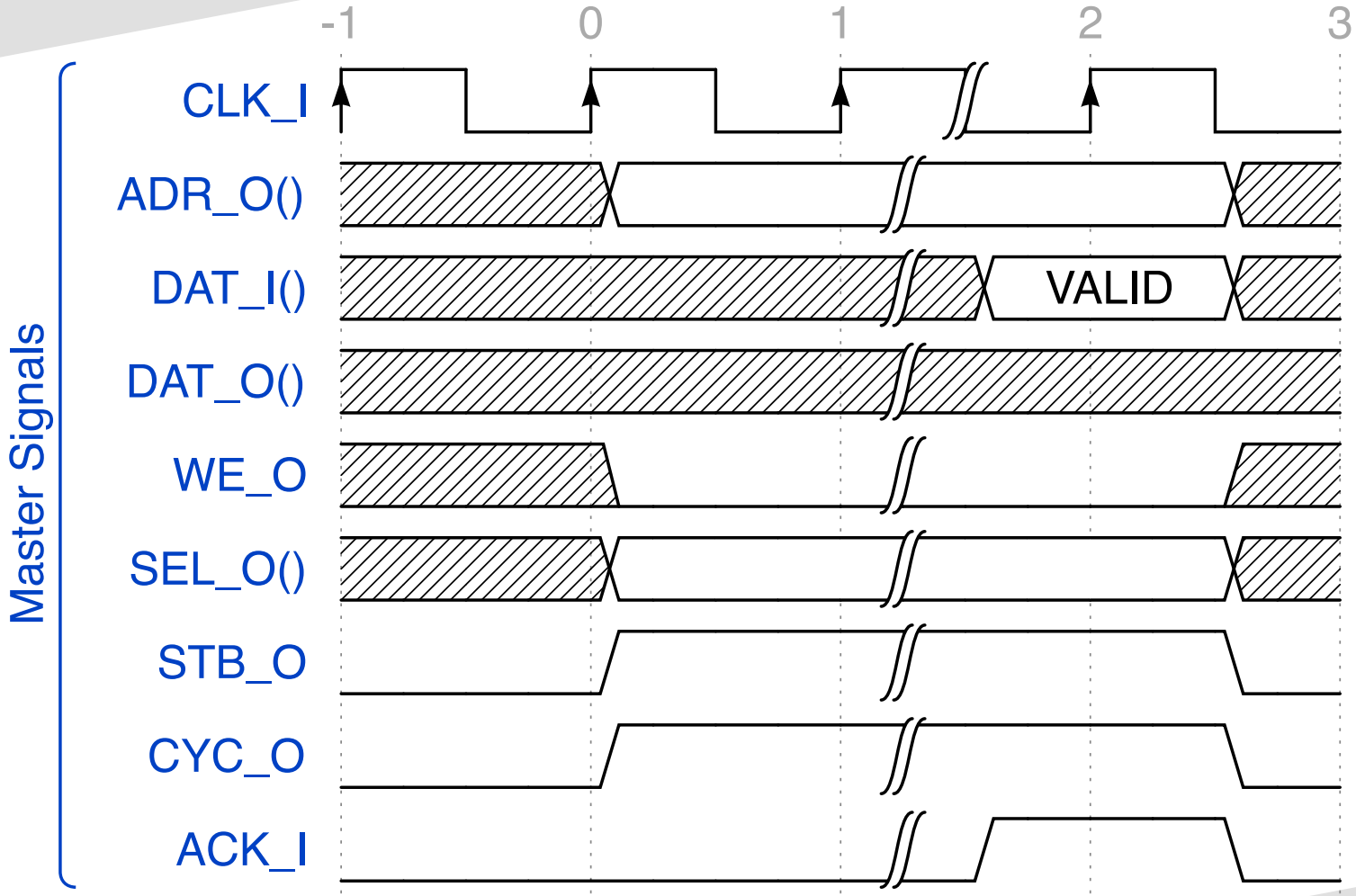
- _I: input
- _O: output
- (): Bus

- Usually you connect XXX_O to XXX_I !

Principle (classic)

- CYC: valid cycle in progress
- STB: valid transfer in progress
  - Usually CYC = STB except for read-modify-write
- DAT: data
- ADR: address
- WE: write enable
- SEL: byte select (for write)
  - Which bytes of the word are written
- ACK: acknowledge
- + RST, CLK

Waveforms

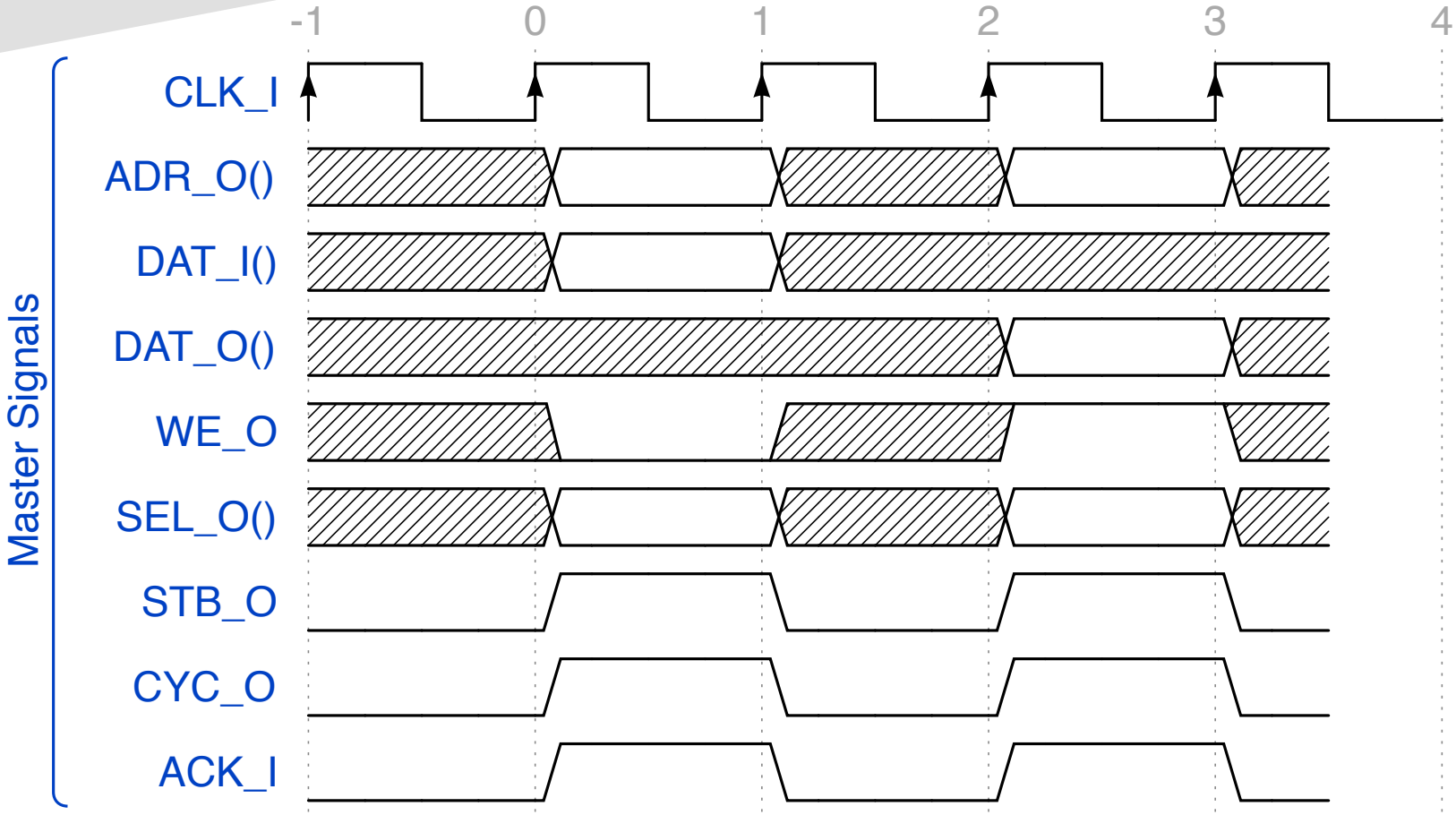

-1
0
1
2
3
Master Signals
CLK_I
ADR_O()
DAT_I()
VALID
DAT_O()
WE_O
SEL_O()
STB_O
CYC_O
ACK_I

The Good

- It’s simple
  - Simplification: make CYC optional
- Could even support asynchronous slaves
  - ACK ← CYC and STB
  - (Permission 3.10)
- Rate control

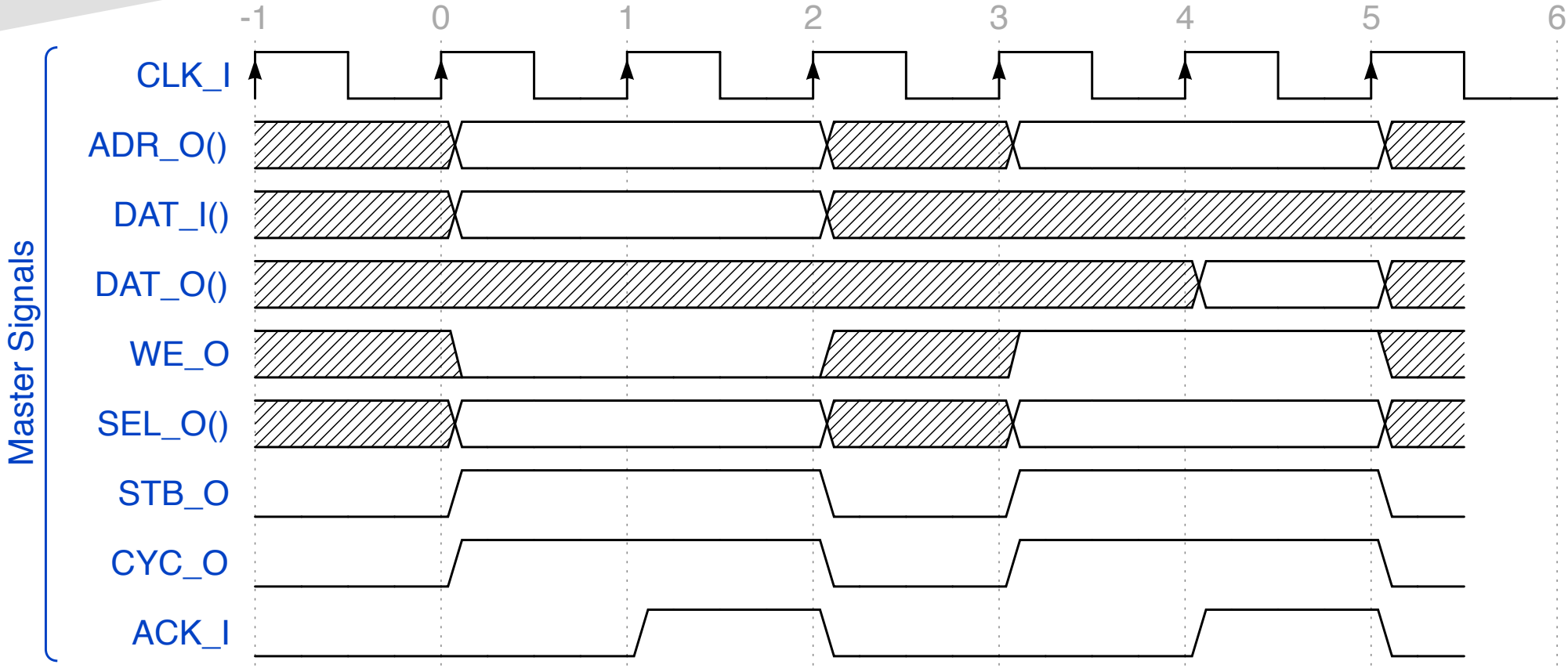
- No license, ...

The Bad



- It’s slow: 50% BW when combinatorial

The Bad



- It’s slow: 33% BW when registered

The Bad

- It’s slow; why ?
- At least two cycles for one transaction
  - 3.1.3 Handshaking Protocol:
  - if it *(ACK)* is asserted, then [STB_O] is negated.

Mitigations:

- Use other wishbone flavours

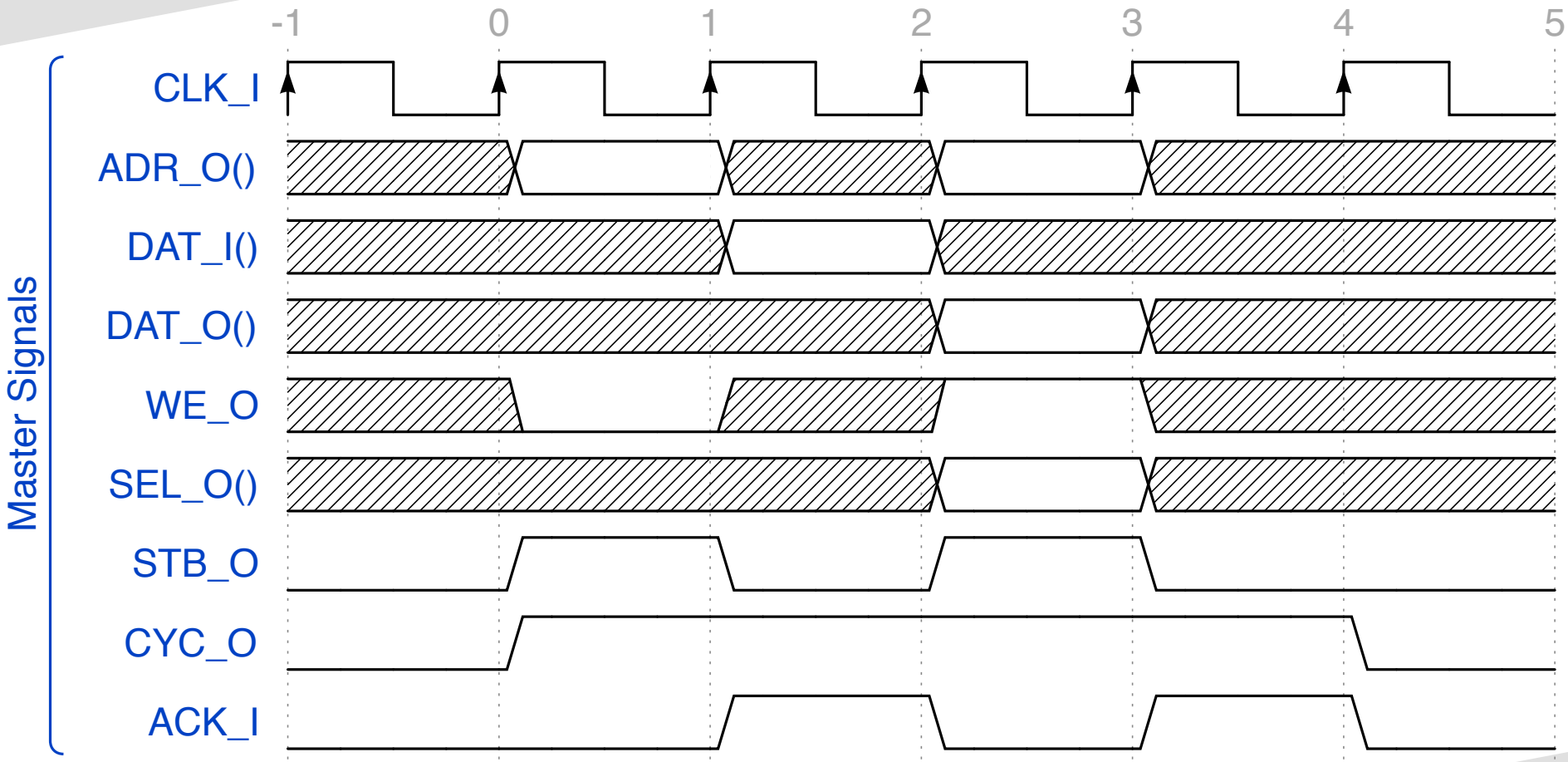
Wishbone flavours

- Wishbone registered feedback
  - Defined by B.3
  - Bursts
  - CTI: cycle type
  - BTE: burst type extension
- More complex
- Use only by OpenRISC CPU ?

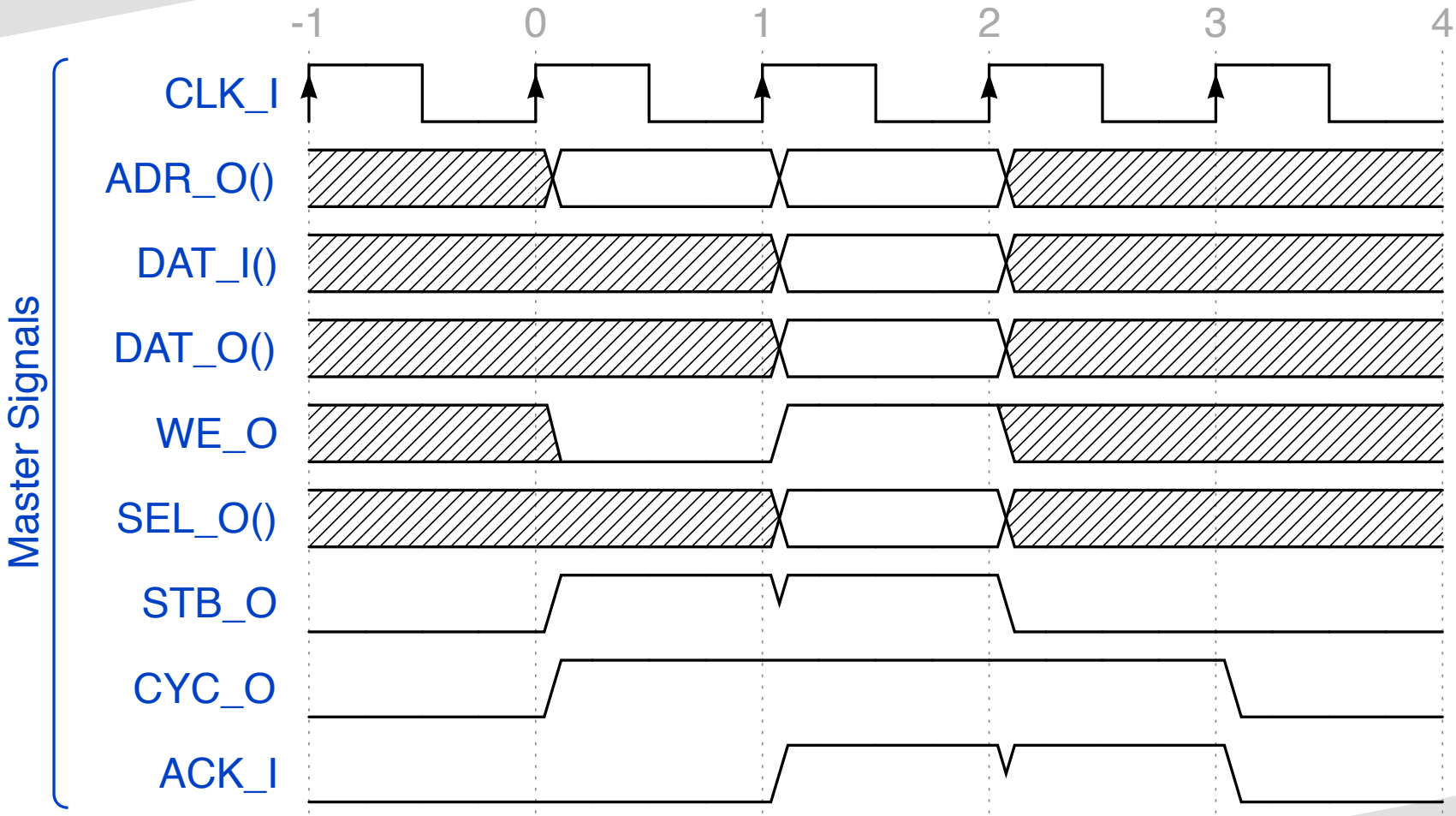
Wishbone flavours

- Wishbone pipelined
  - Defined by B.4
  - STB and ACK are pulses
  - May have multiple transactions in fly
  - New slave output: STALL for throttling
- Almost as simple as classic

Pipelined wishbone



Pipelined wishbone



Assuming STALL is not active

The Bad

- Optional termination signals (in addition to ACK)
  - RTY: retry
  - ERR: error

- What to do if your master doesn’t support them ?
  - (and your slave generate them)
  - ACK_I ← ACK_O or RTY_O or ERR_O ?

The Bad

- The semantic of RTY or ERR varies among buses
  - Does it affect only a beat (wishbone) ?
  - Or the whole burst (AXI) ?

- Makes bridge difficult to write or incorrect

- Conclusion: avoid RTY and ERR

The Hugly (classic)

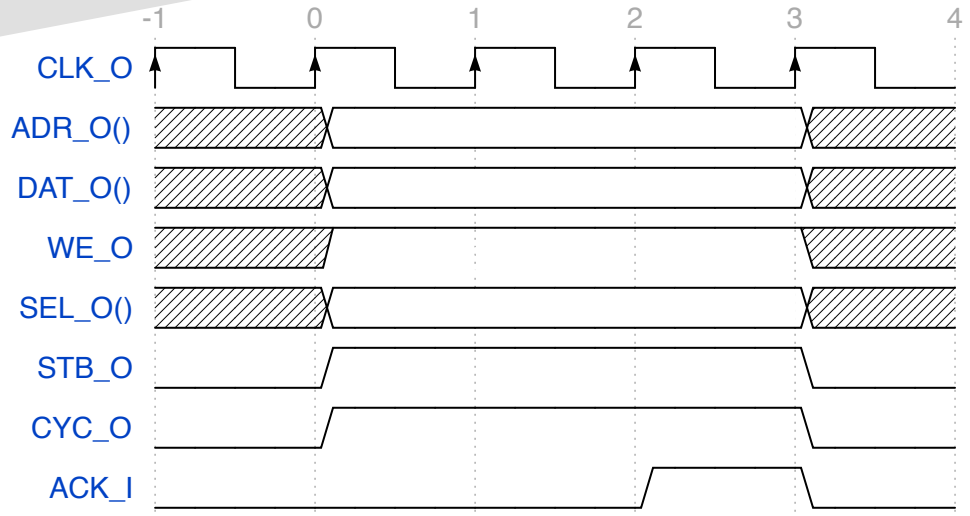
- It is not easy to add a pipeline register

For master outputs:

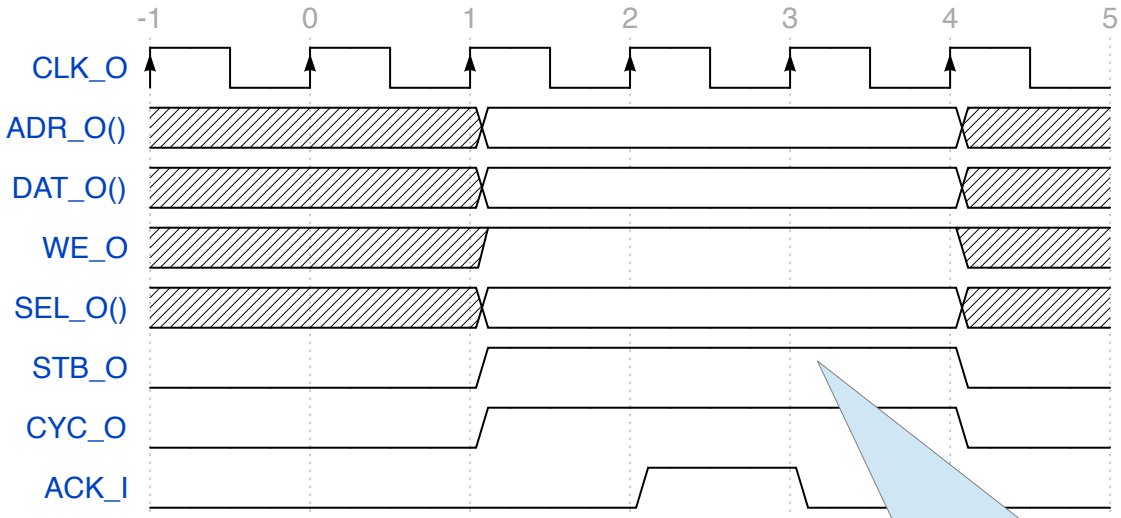
- Violate 3.1.3 Handshaking Protocol:
  - [STB_O] remains asserted until the SLAVE asserts one of the cycle terminating signals [ACK_I]

- Not a problem with the pipelined flavour
  - But need to deal with STALL

Master pipeline



Master



Slave

STB must be deasserted

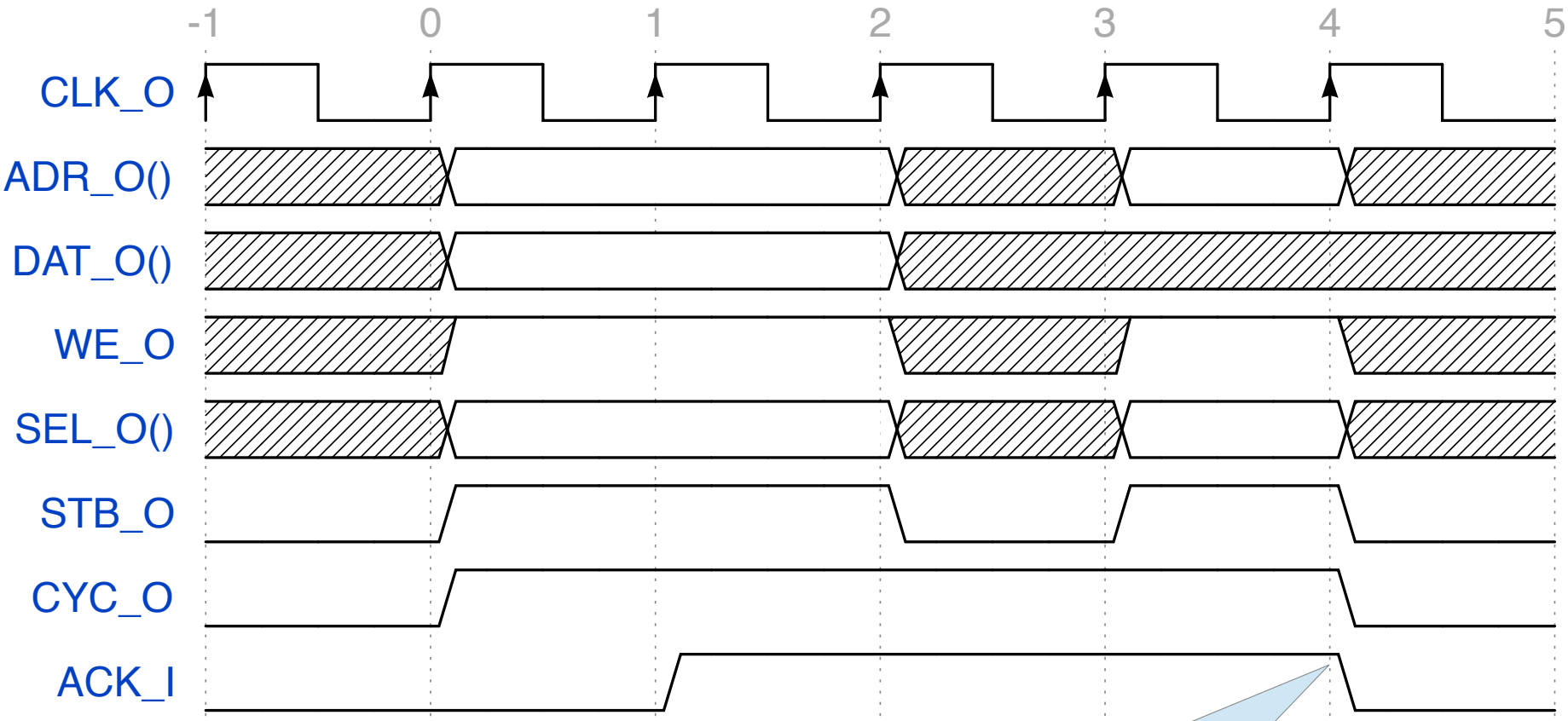
The Hugly (2)

- When does a transaction end ?
  - After ACK, STB has to be negated (3.1.3)
  - At the next cycle
  - But what about ACK ?

The Hugly: ACK

- RULE 3.50
  - SLAVE interfaces MUST be designed so that the [ACK_O], [ERR_O], and [RTY_O] signals are asserted and negated in response to the assertion and negation of [STB_I].
- Doesn’t specify when ACK is asserted or deasserted
- Clearly, ACK can be asserted many cycles later
- So, can it be de-asserted many cycles later ?
  - If so, can STB be activated when ACK is too ?
  - Will ACK terminate the second transaction ?

The Hugly: ACK



ACK for the first
or the second transaction ?

Assert ACK for only 1 cycle,
Combinatorial slaves are OK.

The Hugly: ACK

- Worst:
- PERMISSION 3.35
  - Under certain circumstances SLAVE interfaces MAY be designed to hold [ACK_O] in the asserted state. This situation occurs on point-to-point interfaces where there is a single SLAVE on the interface, and that SLAVE always operates without wait states.
- Eh, I thought it was a ‘MUST’…
- Zero wait states means async slave
- Ok, that’s a simplification

The Hugly: ACK

- Worst in worst:
- RULE 3.55
  - MASTER interfaces MUST be designed to operate normally when the SLAVE interface holds [ACK_I] in the asserted state.
- What does ‘operate normally’ mean ?
- Does the spec tells it must work ?

Conclusion

- Wishbone exists
- It is free
- It works
- It is used!

- Combinatorial slaves are always OK
- Use a single pulse for ACK
- Avoid RTY and ERR

Conclusion

- There is a spec
- Many details and waveforms
- Rules are numbered
- Examples

- Maybe the spec is too long
- Risk of incoherences

Conclusion

- Writing specifications is an art!
- A natural language (English) may not be precise enough
- Formal methods help
  - TODO: add SVA/PSL ?

- Is there a room for a slightly improved SoC bus ?
  - Humm...
- Prefer pulses to state (for easier pipelining)
- Avoid retry or error

Thanks!

“

”