**BUSLAYER SPECIFICATION**

**Read transaction (master)**The module-interface for a read transaction works as follows:

* CLOCK EDGE 0
  + MODULE presents a valid address on [ADDRESS\_I]
  + MODULE negates [WRITE\_I]
  + MODULE presents bank select [BYTE\_SEL\_I]
  + MODULE asserts [REQUEST\_I]
* CLOCK EDGE 1
  + BUSLAYER asserts BUSY\_O in response to asserted [REQUEST\_I],   
    and starts a new bus READ CYCLE (see next section)

Normal termination of a request

* CLOCK EDGE N (N > 1)
  + BUSLAYER asserts READY\_FROM\_BUS\_O
  + BUSLAYER presents valid data in DATA\_FROM\_BUS\_0
  + MODULE negates [REQUEST\_I] in response to asserted [READY\_FROM\_BUS\_O]
* CLOCK EDGE N + 1
  + BUSLAYER negates READY\_FROM\_BUS\_O
  + BUSLAYER negates BUSY\_O

Abnormal termination of a request

* CLOCK EDGE N (N > 1)
  + BUSLAYER asserts ERROR\_FROM\_BUS\_O
  + MODULE negates [REQUEST\_I] in response to asserted [READY\_FROM\_BUS\_O]
* CLOCK EDGE N + 1
  + BUSLAYER negates ERROR\_FROM\_BUS\_O
  + BUSLAYER negates BUSY\_O

**Write transaction (master)**The module-interface for a write transaction works as follows:

* CLOCK EDGE 0
  + MODULE presents a valid address on [ADDRESS\_I]
  + MODULE presents a valid data on [DATA\_TO\_BUS\_I]
  + MODULE asserts [WRITE\_I]
  + MODULE presents bank select [BYTE\_SEL\_I]
  + MODULE asserts [REQUEST\_I]
* CLOCK EDGE 1
  + BUSLAYER asserts BUSY\_O in response to asserted [REQUEST\_I],   
    and starts a new bus WRITE CYCLE (see next section)

Normal termination of a request

* CLOCK EDGE N (N > 1)
  + BUSLAYER asserts READY\_FROM\_BUS\_O
  + MODULE negates [REQUEST\_I] in response to asserted [READY\_FROM\_BUS\_O]
* CLOCK EDGE N + 1
  + BUSLAYER negates READY\_FROM\_BUS\_O
  + BUSLAYER negates BUSY\_O

Abnormal termination of a request

* CLOCK EDGE N (N > 1)
  + BUSLAYER asserts ERROR\_FROM\_BUS\_O
  + MODULE negates [REQUEST\_I] in response to asserted [READY\_FROM\_BUS\_O]
* CLOCK EDGE N + 1
  + BUSLAYER negates ERROR\_FROM\_BUS\_O
  + BUSLAYER negates BUSY\_O

**// control signals  
PSL1: assert always ( (!busy && request) -> next[1](busy) );  
PSL2: assert always ( (!busy && request) -> next[1](wb\_cyc && wb\_stb) );  
PSL3: assert always ( (!wb\_rst && next[1](busy))  
 -> next[1] ((busy) until (ready\_from\_bus || error\_from\_bus)) );**

**PSL4: assert never (ready\_from\_bus && error\_from\_bus);  
PSL5: assert always ( ready\_from\_bus -> next [1] (!ready\_from\_bus && !busy) );  
PSL6: assert always ( error\_from\_bus -> next [1] (!error\_from\_bus && !busy) );**

**// data signals  
PSL10: assert always ( (!busy && request) -> next[1](wb\_we == prev(write, 1)) );  
PSL11: assert always ( (!busy && request) -> next[1](wb\_adr == prev(address, 1)) );  
PSL12: assert always ( (!busy && request) -> next[1](wb\_dat\_o == prev(data\_to\_bus, 1)) );  
PSL13: assert always ( (!busy && request) -> next[1](wb\_sel == prev(byte\_sel, 1)) );**

**Read transaction (slave)**The module-interface for a read transaction works as follows:

* CLOCK EDGE 0
  + BUSLAYER presents a valid address on [ADDRESS\_O]
  + BUSLAYER negates [WRITE\_O]
  + BUSLAYER presents bank select [BYTE\_SEL\_O]
  + BUSLAYER asserts [REQUEST\_O]

Normal termination of a request

* CLOCK EDGE N (N >= 0)
  + MODULE asserts [DONE\_I]
  + MODULE presents valid data on [DATA\_TO\_BUS\_I]
* CLOCK EDGE N + 1
  + BUSLAYER negates [REQUEST\_O]
  + MODULE negates [DONE\_I]

Abnormal termination of a request

* CLOCK EDGE N (N >= 0)
  + MODULE asserts [ERR\_I]
* CLOCK EDGE N + 1
  + BUSLAYER negates [REQUEST\_O]
  + MODULE negates [ERR\_I]

**Read transaction (write)**The module-interface for a read transaction works as follows:

* CLOCK EDGE 0
  + BUSLAYER presents a valid address on [ADDRESS\_O]
  + BUSLAYER presents a valid data on [DATA\_FROM\_BUS\_O]
  + BUSLAYER asserts [WRITE\_O]
  + BUSLAYER presents bank select [BYTE\_SEL\_O]
  + BUSLAYER asserts [REQUEST\_O]

Normal termination of a request

* CLOCK EDGE N (N >= 0)
  + MODULE asserts [DONE\_I]
* CLOCK EDGE N + 1
  + BUSLAYER negates [REQUEST\_O]
  + MODULE negates [DONE\_I]

Abnormal termination of a request

* CLOCK EDGE N (N >= 0)
  + MODULE asserts [ERR\_I]
* CLOCK EDGE N + 1
  + BUSLAYER negates [REQUEST\_O]
  + MODULE negates [ERR\_I]

**BUS-INTERFACE**

**Wishbone bus read cycle**

The bus protocol works as follows:

* CLOCK EDGE 0
  + MASTER presents a valid address on [ADR\_O]
  + MASTER negates [WE\_O]
  + MASTER presents bank select [SEL\_O]
  + MASTER asserts [CYC\_O] and [STB\_O]

SLAVE presents valid data immediately

* CLOCK EDGE 1
  + responding SLAVE presents valid data on [DAT\_I]
  + responding SLAVE asserts [ACK\_I] in response to [STB\_O] to indicate valid data
* CLOCK EDGE 2
  + MASTER latches data on [DAT\_I]
  + MASTER negates [STB\_O] and [CYC\_O] to indicate the end of the cycle
  + Responding SLAVE negates [ACK\_I] in response to negated [STB \_O]

SLAVE does not present valid data immediately

* CLOCK EDGE 1
  + responding SLAVE asserts [STALL\_I]
  + MASTER negates [STB\_O] in response to asserted [STALL\_I]
* CLOCK EDGE N
  + responding SLAVE presents valid data on [DAT\_I]
  + responding SLAVE negates [STALL\_I] and asserts [ACK\_I] to indicate valid data
* CLOCK EDGE N+1
  + MASTER latches data on [DAT\_I]
  + MASTER negates [CYC\_O] to indicate the end of the cycle
  + responding SLAVE negates [ACK\_I] in response to negated [CYC\_O]

SLAVE cannot present valid data

* CLOCK EDGE 1
  + responding SLAVE asserts [ERR\_I] in response to [STB\_O] to indicate not available data
* CLOCK EDGE 2
  + MASTER negates [STB\_O] and [CYC\_O] to indicate the end of the cycle
  + Responding SLAVE negates [ERR\_I] in response to negated [STB \_O]

**Wishbone bus write cycle**

The bus protocol works as follows:

* CLOCK EDGE 0
  + MASTER presents a valid address on [ADR\_O]
  + MASTER presents a valid data on [DAT\_O]
  + MASTER asserts [WE\_O]
  + MASTER presents bank select [SEL\_O]
  + MASTER asserts [CYC\_O] and [STB\_O]

SLAVE consumes data immediately

* CLOCK EDGE 1
  + responding SLAVE asserts [ACK\_I]
* CLOCK EDGE 2
  + MASTER negates [STB\_O] and [CYC\_O] to indicate the end of the cycle
  + Responding SLAVE negates [ACK\_I] in response to negated [STB \_O]

SLAVE does not consume data immediately

* CLOCK EDGE 1
  + responding SLAVE asserts [STALL\_I]
  + MASTER negates [STB\_O] in response to asserted [STALL\_I]
* CLOCK EDGE N
  + responding SLAVE negates [STALL\_I] and asserts [ACK\_I] to indicate data was accepted
* CLOCK EDGE N+1
  + MASTER negates [CYC\_O] to indicate the end of the cycle
  + responding SLAVE negates [ACK\_I] in response to negated [CYC\_O]

SLAVE cannot consume data

* CLOCK EDGE 1
  + responding SLAVE asserts [ERR\_I] in response to [STB\_O]
* CLOCK EDGE 2
  + MASTER negates [STB\_O] and [CYC\_O] to indicate the end of the cycle
  + Responding SLAVE negates [ERR\_I] in response to negated [STB \_O]