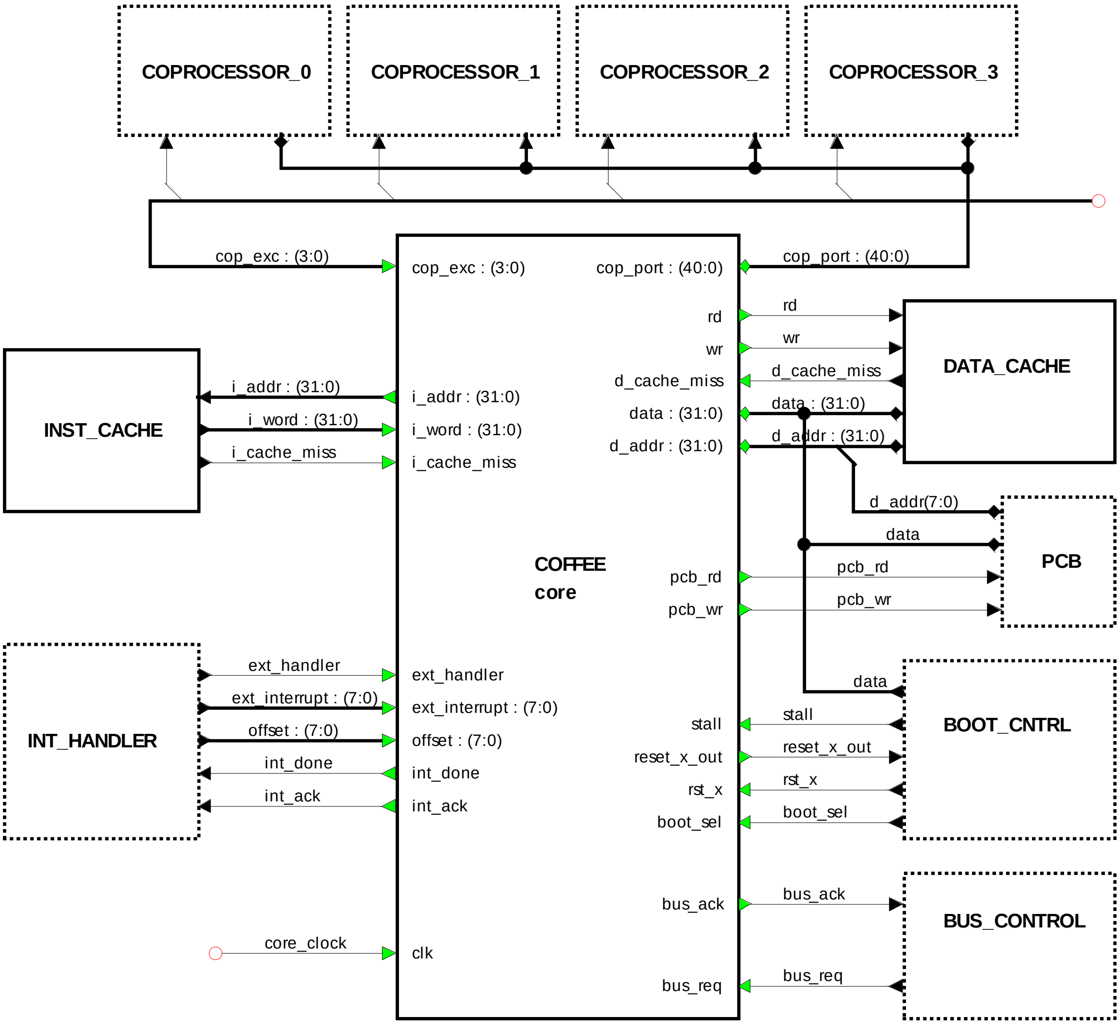
Interface specification of the COFFEE RISC Core



Picture 1, Interfacing core.

1 General

Picture 1 shows an example of interfacing the core. This is not the only possible way to connect to core. Optional peripherals are drawn with dashed line. External interrupt handler, boot agent, PCB (peripheral control block) and the coprocessors are optional. Also the use of bus_req and bus_ack signals is optional. bus_req and bus_ack –signals allow sharing the data bus. Boot agent can be used if the boot address has to be determined externally. PCB is a user defined block to interface peripheral devices directly. It can have, for example, configuration registers mapped to some of the memory addresses. PCB address space is defined by software. Unused inputs should be driven to a state defined in port specification.

Definitions and assumptions used in this document

The rising edge of the clock signal is the sampling instant unless otherwise specified. Some interfacing signals are static while others are pulse type signals. Time delays are defined in chapter ‘Timing specification’. Note the distinction between signals and ports. An asynchronous signal is not the same as an asynchronous port! Note that an asynchronous port can always be driven by a synchronous signal if the timing constraints are fulfilled.

Pulse type signal

With pulse type signals an event is signalled by first driving the signal to active state and after one or more clock cycles driving it to inactive state. The pulse length varies depending on activities on pipeline. For example, having to wait for a memory access can extend a pulse to multiple clock cycles long. This kind of signal will be active on consecutive clock edges even though signalling just one event.

Static signal

Each time a static signal is high on the active clock edge, an event is signalled. Static signal may or may not go inactive between consecutive events depending on timing of the events. If the signal is active on consecutive clock edges then consecutive events are signalled.

Asynchronous signal(later referred as AS):

A signal which is evaluated in the same cycle as the inputs change. Typically signals which have to react to a certain input condition immediately (in one clock cycle) are asynchronous. Also signals which come from a different clock domain are asynchronous. Both types of signals have differerent timing specifications.

Synchronous signal(later referred as SS):

A signal which changes its value slightly after the active edge of the clock. Typically a signal directly from an output of a flip flop (maximum of few gates after a flip flop).

Asynchronous input port(later referred as AIP):

Input which is not sampled on clock edge (goes directly to logic). Must be driven to a valid state time T_{sx} before the active edge of the clock or time T_{dx} after a change in input conditions (outputs from the core).

Synchronous input port(later referred as SIP):

An input which is sampled on active clock edge. The input must be valid time T_s before the active clock edge or time T_{dx} after a change in input conditions (outputs from the core).

Synchronized input port(later referred as SZIP):

An input which uses a special synchronizer circuit (usually adds delay). The input can change at any moment but it must be held constant for a specified time.

Asynchronous output port(later referred as AOP):

An output which is valid after an arbitrary propagation time from the change of inputs. This design does not have any asynchronous outputs.

Synchronous output port(later referred as SOP):

A port which drives a synchronous signal. See *synchronous signal*.

2 Interface descriptions

2.1 Interfacing instruction cache/instruction memory

Table 2.1, Instruction cache interfacing signals.

signal	direction	purpose/description when active	port type
I_CACHE_MISS	in	Instruction cache signals to the core to wait in case of a miss.	AIP
I_WORD[31..0]	in	Data from the instruction cache	SIP
I_ADDR[31..0]	out	Address of the requested word from the instruction cache	SOP

Notes:

I_CACHE_MISS –signal must be evaluated in less than one clock cycle time. If this is not possible, synchronization structures which prevent signal transition during rising clock edge must be used. Memory access time can be up to sixteen clock cycles long. The data on **I_WORD** –bus must be valid when specified amount of wait cycles has elapsed from asserting a new address on **I_ADDR** –bus. The number of wait cycles can be configured by software. See timing specification.

2.2 Interfacing data cache

Table 2.2, Data cache interfacing signals.

signal	direction	purpose/description when active	type
D_CACHE_MISS	in	Data cache signals to the core to wait in case of a miss.	AIP
DATA[31..0]	inout	Data to/from the Data cache/other device. (or boot address, see Port descriptions)	SIP[1] SOP
D_ADDR[31..0]	inout	Address of the accessed item in data memory.	SIP[1] SOP
WR	out	Data cache write signal. Active when high.	SOP

RD	out	Data cache read signal. Active when high.	SOP
BUS_REQ	in	External device can request the bus by asserting this signal. See document 'Coffee shared data bus'	AIP
BUS_ACK	out	The core reponds to bus request by asserting this signal when the bus is free.	SOP

[1] Bus behaviour can be asynchronous since the sampling circuitry is isolated from the core logic until data is latched in.

Notes:

D_CACHE_MISS –signal must be evaluated in less than one clock cycle time. If this is not possible, synchronization structures which prevent signal transition during rising clock edge must be used. Memory access time can be up to sixteen clock cycles long. The data on **DATA** –bus must be valid when specified amount of wait cycles has elapsed from starting a new access. The number of wait cycles can be configured by software. See timing specification.

Signals **BUS_REQ** and **BUS_ACK** doesn't have to be used if the bus is not shared with devices which are communicating directly with each other (for example, DMA). See document 'COFFEE shared data bus' for details about shared bus.

D_ADDR might not be aligned to word boundary, that is, it might not be divisible by four. Depending on the implementation, bits 1 downto 0 of **D_ADDR** can be used for example as chip selects to allow byte addressing. It is even possible to have a 4 GIGAWORD address space instead of 4 GB space, if each address corresponds to a 32 bit word.

2.3 Interfacing coprocessors

Table 2.3, Coprocessor interfacing signals.

signal	**Direction**	**purpose/description when active**	**type**
COP_EXC[3..0]: COP_EXC(3) – COP 3 COP_EXC(2) – COP 2 COP_EXC(1) – COP 1 COP_EXC(0) – COP 0	In	Coprocessor exception. Coprocessor can interrupt the core by driving a pulse on this input. Sensitive to falling edge.	SZIP
COP_PORT(40): **WR_COP**	out	Write to cop. Write access to coprosessor register file.	SOP
COP_PORT(39): **RD_COP**	out	Read from cop. Read access to coprosessor register file.	SOP
COP_PORT[38..37]: **C_INDX**	out	Coprocessor index used to address one of the four possible coprocessors	SOP

COP_PORT[36..32]: R_INDX	out	Register index used to select the right register from the accessed coprocessor register bank.	SOP
COP_PORT[31..0]: DATA	inout	Data to/from the coprocessor.	SOP SIP[1]

[1] Bus behaviour can be asynchronous since the sampling circuitry is isolated from the core logic until data is latched in from coprocessor data bus.
See timing specification.

Notes:

As the memory interfaces, also coprocessor interface can be configured by software to use a fixed amount of wait cycles. This should be considered if the coprocessor uses a different clock which is slower than the clock of the core. Note, that input signals from coprocessors can be asynchronous as long as they are evaluated inside specified time windows, see timing specification at the end of this document. General **STALL** –signal can be used to freeze the core if needed, but its use shoud be avoided because of performance penalty.

2.4 Interfacing external interrupt handler

Table 2.4, External interrupt handler interfacing signals.

signal	direction	purpose/description when active	type
EXT_HANDLER/ SYNC_EN_X	in	When high, core assumes that an external interrupt handler is present and uses **OFFSET** – signal when calculating handler routine address. When low, offset is ignored and synchronization circuitry enabled. See document about interrupts.	AIP
OFFSET[7..0]	in	An offset used when calculating the starting address of an interrupt service routine. Used only if **EXT_HANDLER** – signal is active. See notes below.	SIP
EXT_INTERRUPT [7..0]	in	Signals from interrupt sources or from the external interrupt handler, if present. Each signal corresponds to one interrupt source. The input is sensitive to a falling edge of the signal. See notes below.	SZIP/SIP
INT_ACK	out	The core signals that the latest interrupt request has been accepted and the service routine has just started.	SOP
INT_DONE	out	The core signals that the an interrupt service routine has finished.	SOP

Notes:

The timing of the EXT_INTERRUPT signal depends on the EXT_HANDLER signal. See timing specification.

The interrupt address is calculated as follows if EXT_HANDLER – input is pulled high:

base_addr[31 downto 12] & **OFFSET[7 downto 0]** & 0000,
where '&' means concatenation.

The **base_addr[31 downto 12]** equals the upper 20 bits of the value in the corresponding interrupt register (interrupt vector, see registers: CCB).

An interrupt request is saved internally (INT_PEND –register) and will be overwritten if a new request is signalled before the previous one was served. If interrupts are disabled for a long time this can be the case.

2.5 Interfacing peripheral control block

Peripheral control block is a memory mapped module used to interface and control some of the peripherals of the core directly. It is connected to the same data and address buses than data cache (if present). Interfacing is similar than for data cache.

Table 2.5, PCB interfacing signals

signal	Direction	purpose/description when active	type
DATA[31..0]	Inout	Data to/from the PCB (or data memory).	SIP SOP
D_ADDR[31..0]	Inout	Address of the accessed item in PCB(or data memory)	SIP SOP
PCB_WR	Out	PCB write signal. Active when high.	SOP
PCB_RD	Out	PCB read signal. Active when high.	SOP

Notes:

The wait cycle setting for data memory access applies to PCB access as well.

2.6 Other interface signals

Table 2.6, Other interfacing signals

Signal	Direction	Purpose/description when active	type
RST_X	In	Asynchronous active low reset signal. Synchronized inside core. See port descriptions	SZIP
CLK	In	Core clock.	
BOOT_SEL	In	When driven high, boot address is read from the data bus. When low, core will boot at 00000000h.	AIP
STALL/STROBE	In	General stall input. Can be used to freeze the pipeline of the core. Does not disable timers neither wait cycle counters. Used as boot address strobe if BOOT_SEL is tied high. See timing specification.	AIP
RESET_X_OUT	Out	Synchronised version of RST_X –signal. Pulses low also when internal watchdog reset occurs.	SOP

Notes:
If boot address should be determined by external logic pull BOOT_SEL high permanently, otherwise pull it low.

3 Port descriptions

All unused inputs should be driven to inactive state defined in the table below.

Table 3.1, Port descriptions

Name of the port	dir	active state	inactive state	type	reset state [1]
BUS_REQ	in	high	low	AIP	Must be driven to a valid state
BUS_ACK	out	high	low	SOP	Low
COP_EXC[3..0]	in	high (sensitive to falling edge)	low	SZIP	Must be driven to a valid state
COP_PORT(39) **RD_COP**	out	high	low	SOP	Low
COP_PORT(40) WR_COP	out	high	low	SOP	Low
COP_PORT[31..0] **DATA**	inout	-	-	SOP SIP	Floating
COP_PORT[36..32] **R_INDX**	out	-	-	SOP	all zeros
COP_PORT[38..37]	out	-	-	SOP	all zeros

C_INDX					
D_ADDR[31..0]	inout	-	-	SOP SIP	Floating
D_CACHE_MISS	in	high	low	AIP	Must be driven to a valid state
DATA[31..0]	inout	-	-	SOP SIP	Floated by the core. A valid boot address must be driven on bus by external reset logic if BOOT_SEL is active. [2]
EXT_HANDLER	in	high	low	AIP	Must be driven to a valid state
INT_ACK	out	high	low	SOP	low
INT_DONE	out	high	low	SOP	low
EXT_INTERRUPT[7..0]	in	high (sensitive to falling edge)	low	SZIP /SIP	Must be driven to a valid state
I_ADDR[31..0]	out	-	-	SOP	Boot address after few clock cycles from asserting reset.
I_CACHE_MISS	in	high	low	AIP	Must be driven to a valid state.
I_WORD[31..0]	in	-	-	SIP	-
OFFSET[7..0]	in	-	-	SIP	Must be driven to a valid state
RD	out	high	low	SOP	Low
WR	out	high	low	SOP	Low
PCB_RD	out	high	low	SOP	Low
PCB_WR	out	high	low	SOP	Low
RST_X	in	low	high	SZIP	See timing specification.
CLK	in	rising edge			Clk must have settled before releasing reset.
BOOT_SEL	in	high	low	AIP	Must be driven to a valid state.
STALL	in	high	low	AIP	Must be driven to a valid state. Dual function. See timing specification.
RESET_X_OUT	out	low	high	SOP	Follows RST_X – input. See timing specification.

[1] Values which should be driven on inputs (and is driven on outputs) when reset is active (low). After releasing reset, normal values should appear on outputs after two or three clock cycles depending on the timing of the asynchronous reset signal.

[2] When asserting rst_x signal external boot logic has to drive a valid address on data bus if enabled by BOOT_SEL. The address is used as the starting value for program counter. See timing specification.

4 Timing specification

Diagrams T.1 through T.14 in chapter 4 illustrate timing of COFFEE interfacing signals. Conformance to this timing specification should always be checked after synthesis. Table 4.1 explains markings used in figures.

4.1 Instruction memory interface timing

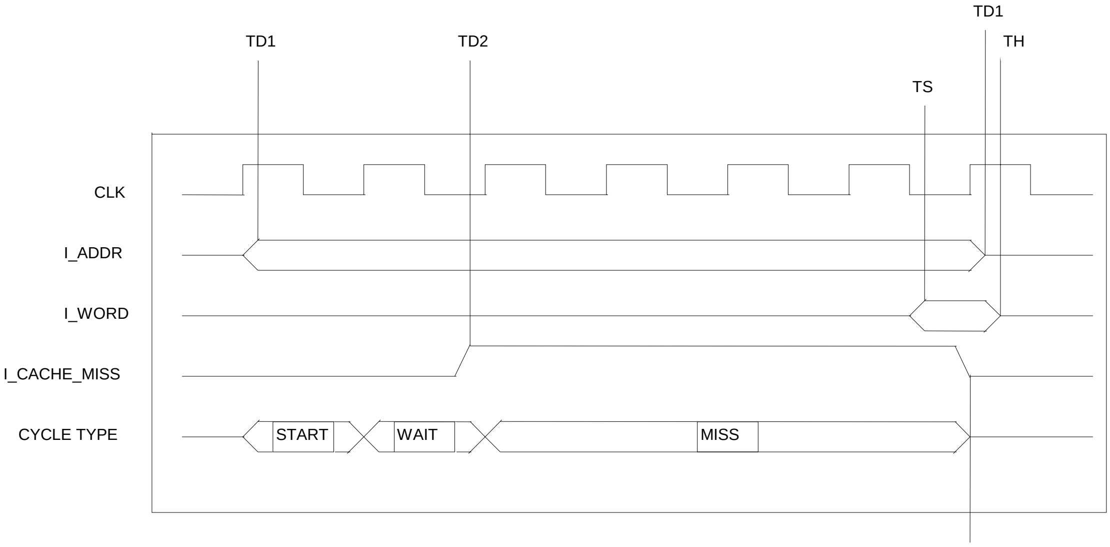


FIGURE T.4, INSTRUCTION FETCH

4.2 Data memory interface timing

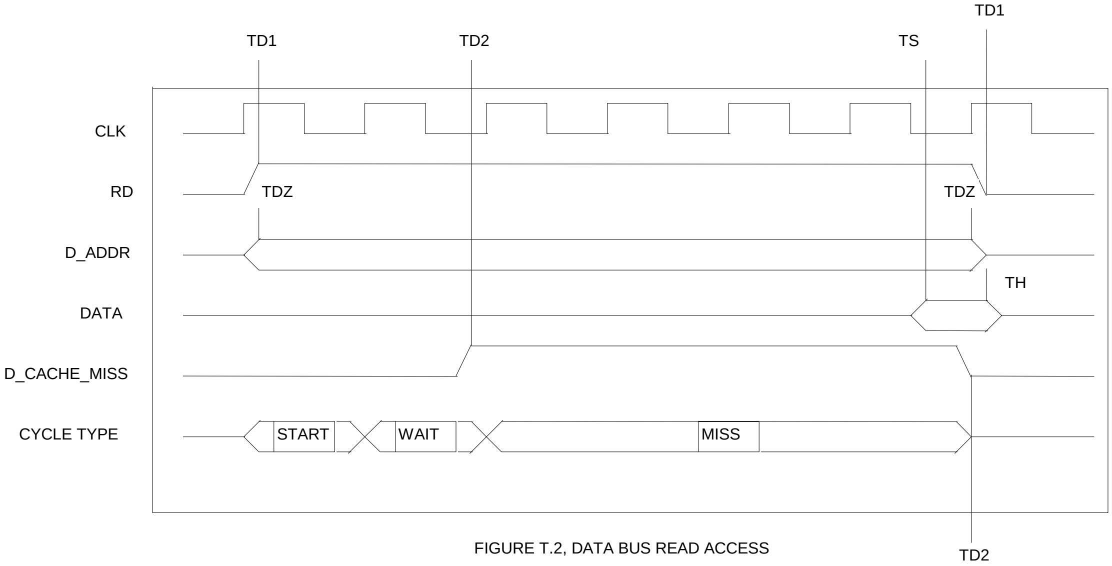


FIGURE T.2, DATA BUS READ ACCESS

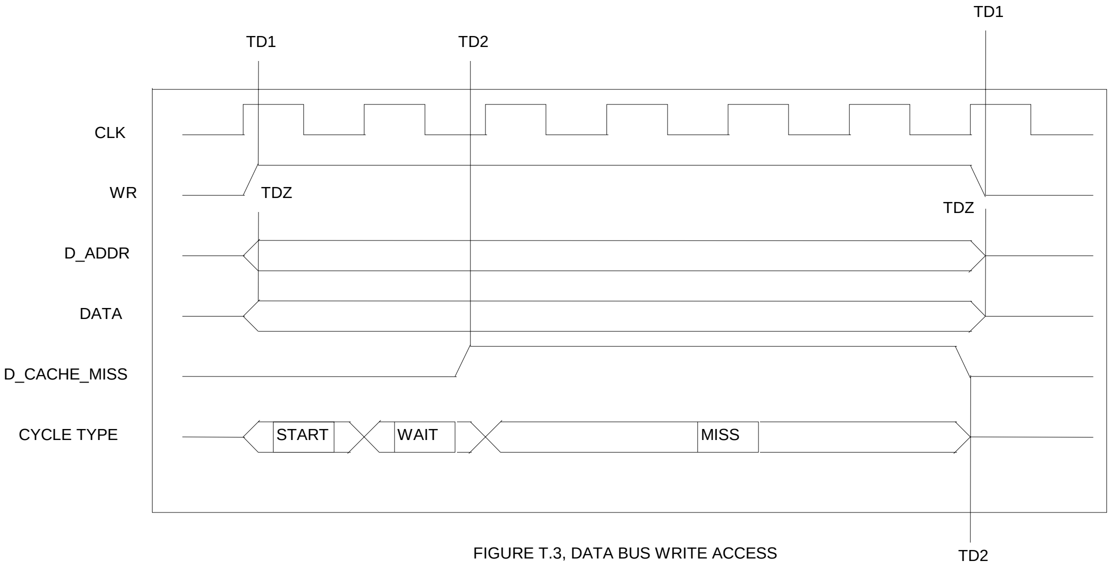


FIGURE T.3, DATA BUS WRITE ACCESS

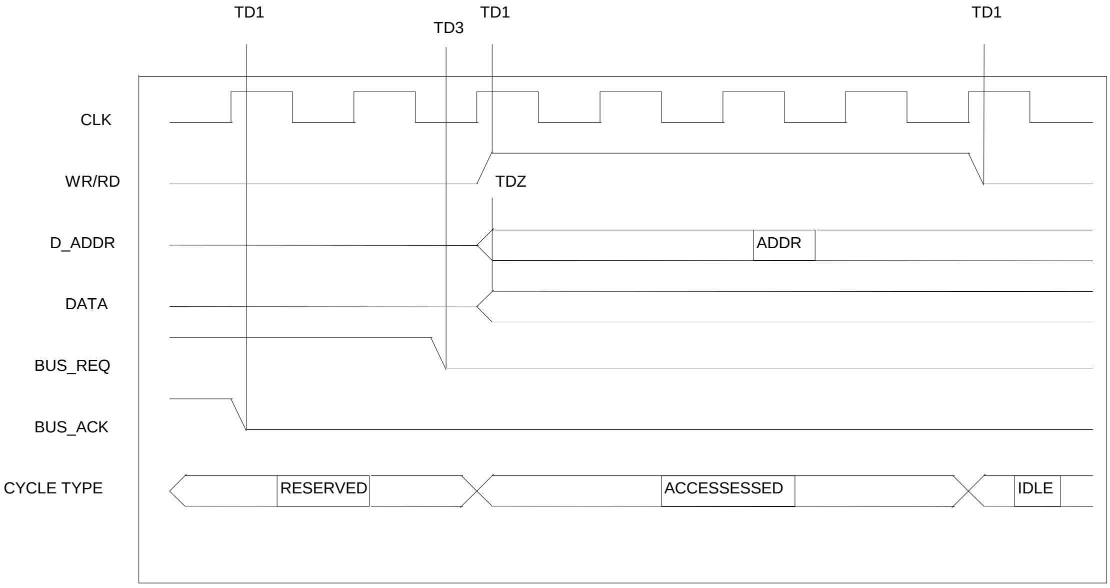


FIGURE T.5, DATA BUS STATE TRANSITION: RESERVED => ACCESSED

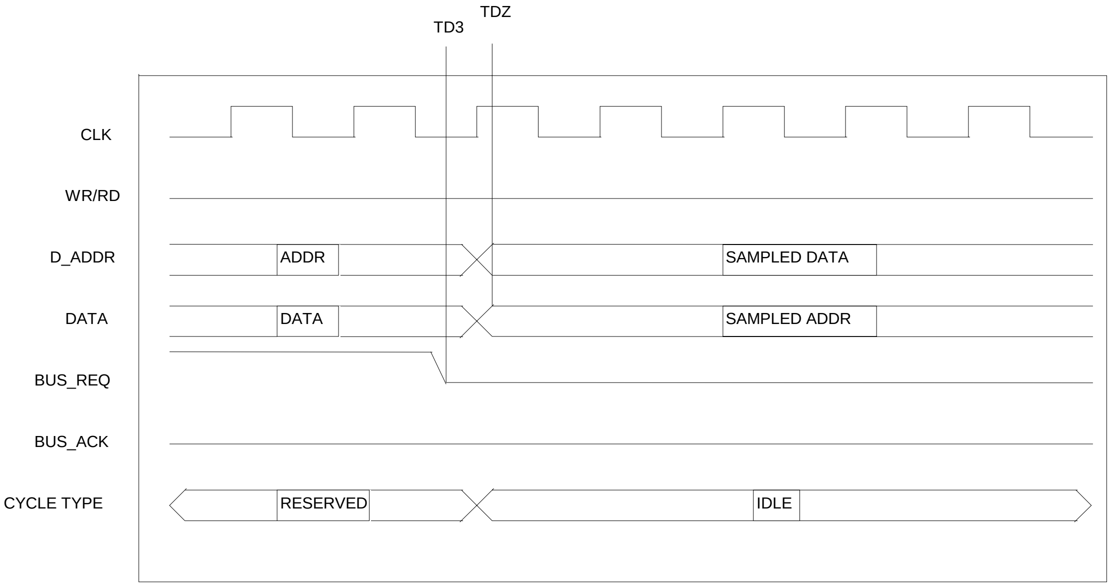


FIGURE T.6, DATA BUS STATE TRANSITION: RESERVED => IDLE

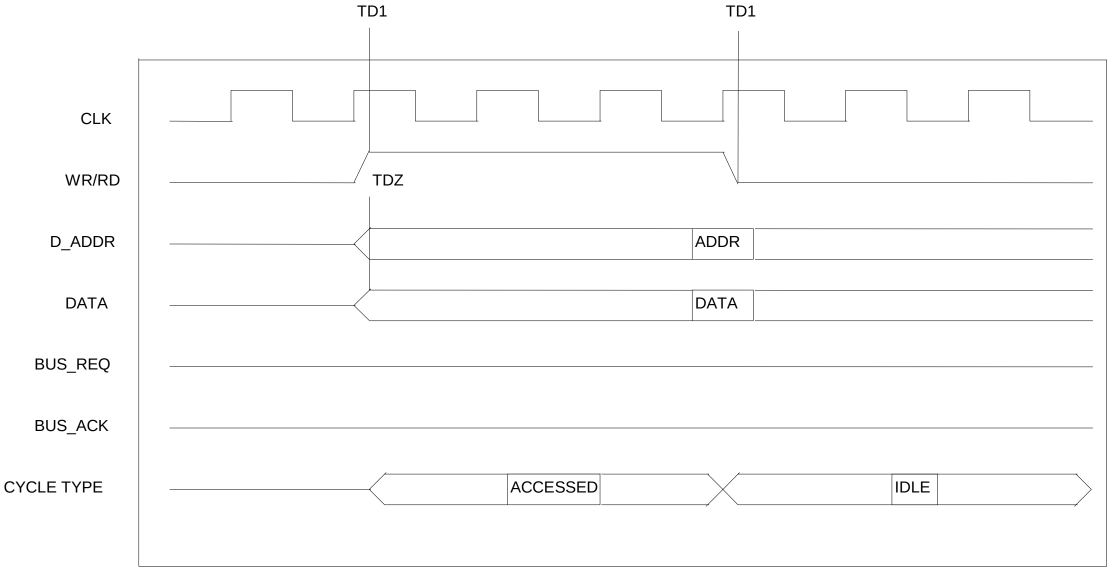


FIGURE T.7, DATA BUS STATE TRANSITION: ACCESSED => IDLE

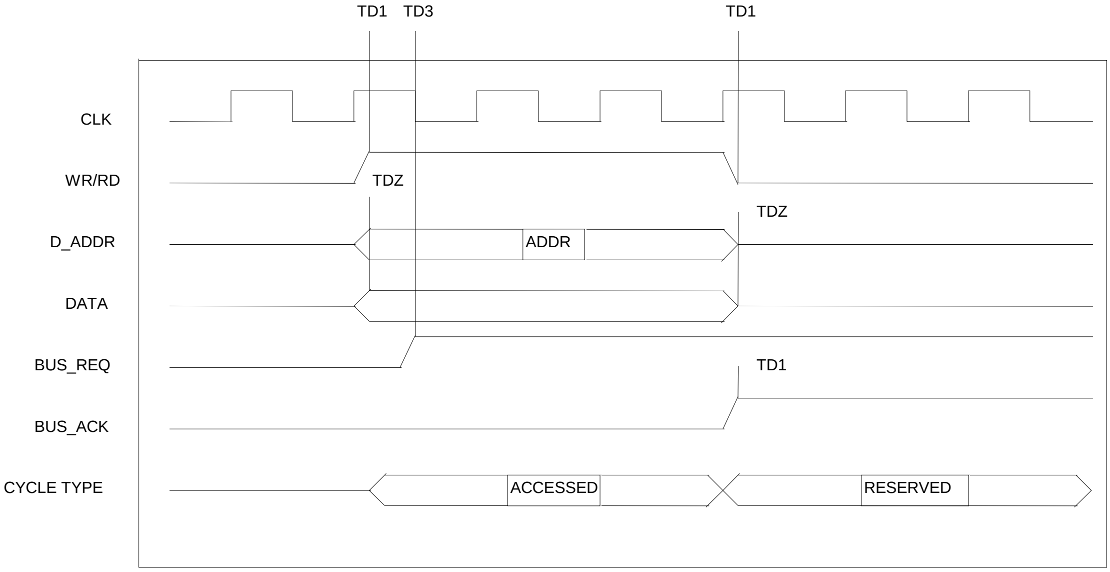


FIGURE T.8, DATA BUS STATE TRANSITION: ACCESSED => RESERVED

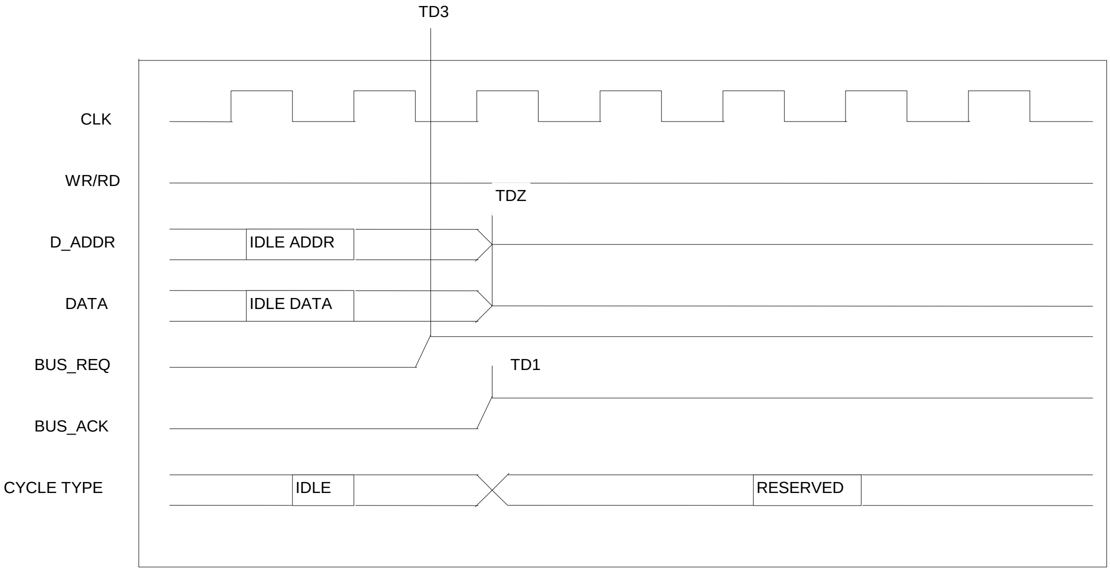


FIGURE T.9, DATA BUS STATE TRANSITION: IDLE => RESERVED

4.3 Coprocessor interface timing

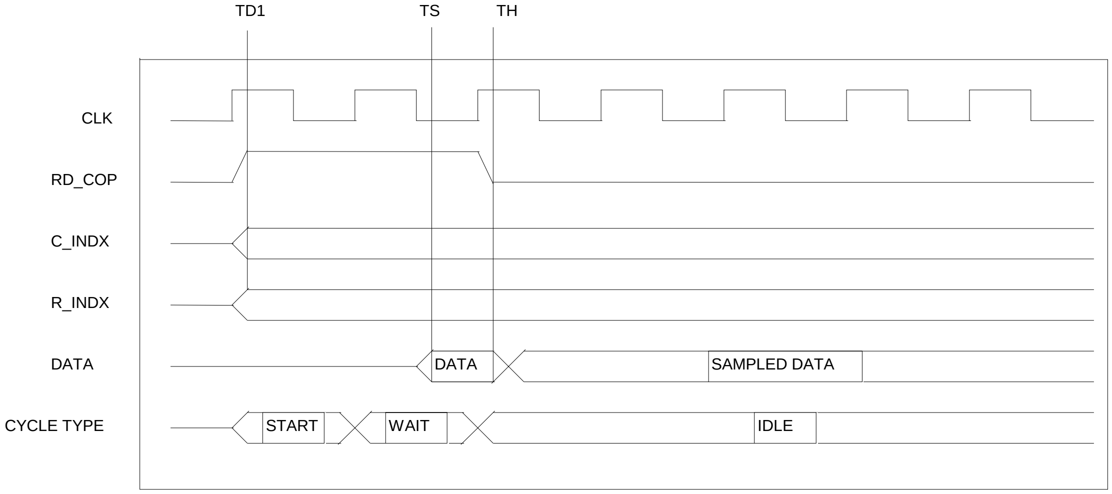


FIGURE T.10, COPROCESSOR READ ACCESS

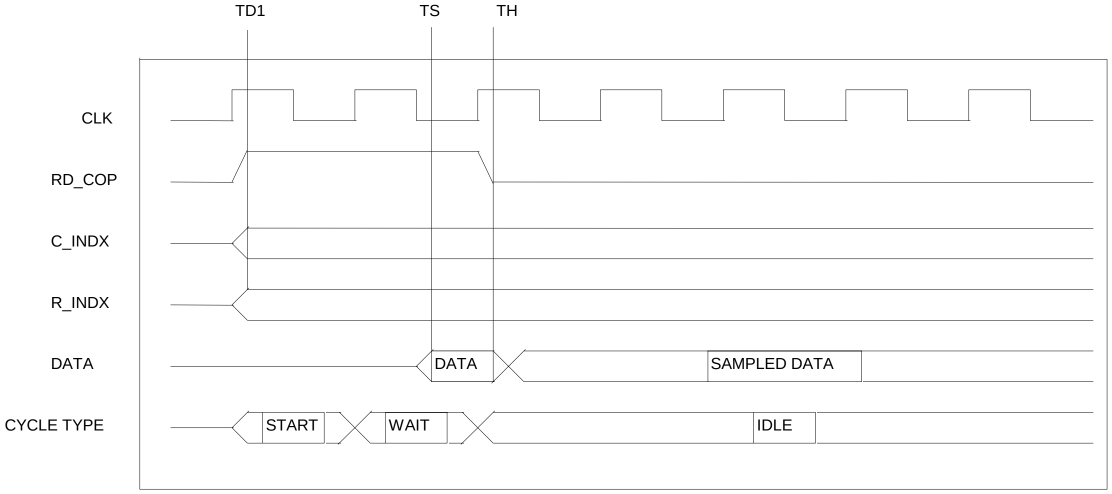


FIGURE T.10, COPROCESSOR READ ACCESS

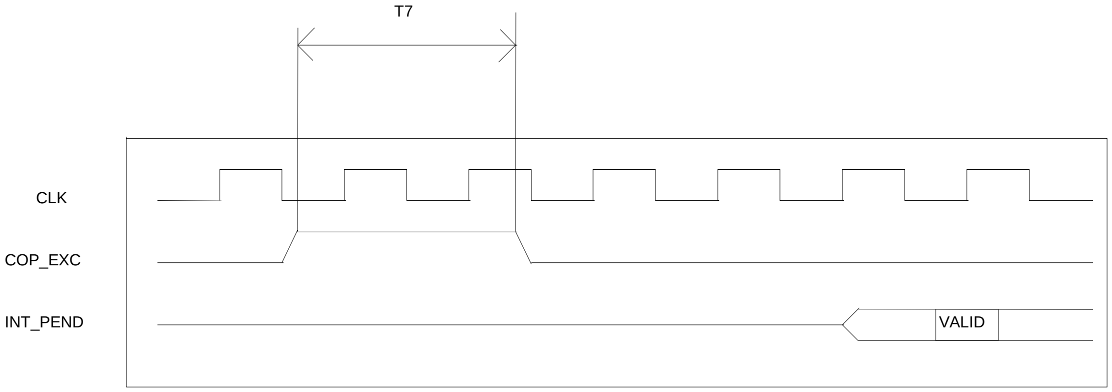


FIGURE T.14, COP_EXC -SIGNAL TIMING

4.4 Interrupt signal timing

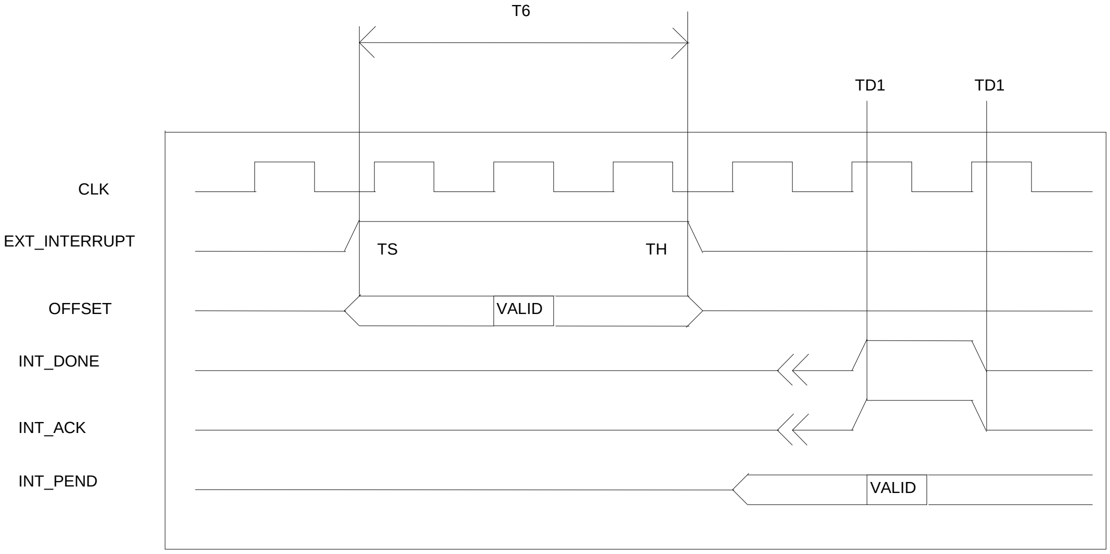


FIGURE T.12, INTERRUPT SIGNAL TIMING WHEN EXT_HANDLER/SYNC_EN_X -INPUT IS PULLED HIGH

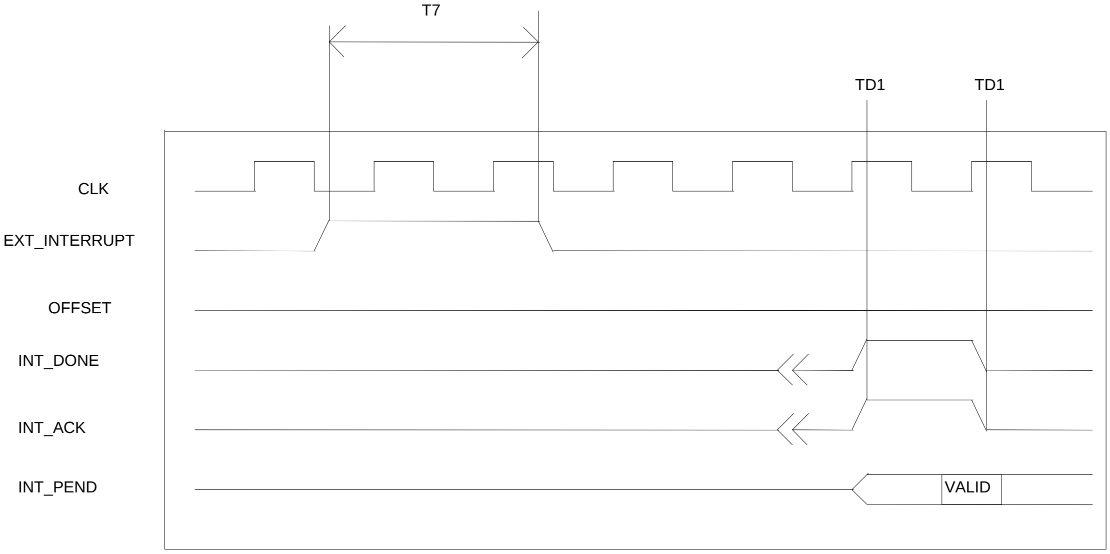


FIGURE T.13, INTERRUPT SIGNAL TIMING WHEN EXT_HANDLER/SYNC_EN_X -INPUT IS PULLED LOW

4.5 Reset signal timing & Timing key

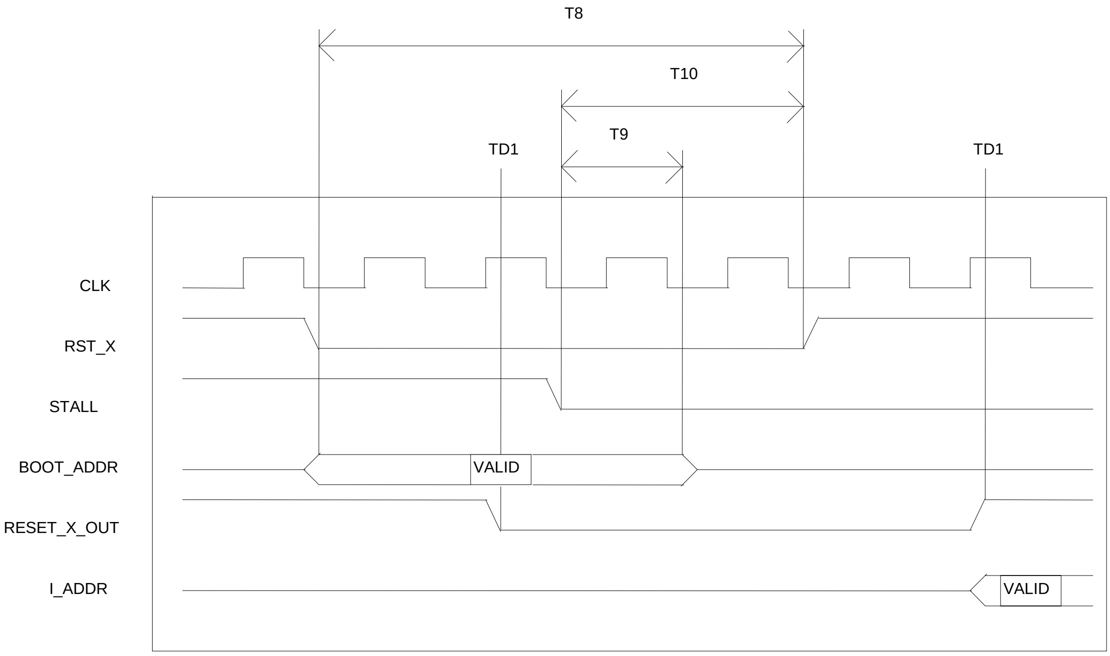


FIGURE T.1, SIGNAL TIMING AT RESET

Figure T.1 above illustrates signal timing when boot address is provided externally via data bus. Latching boot address from data bus is controlled by dual function input STALL/STROBE. A falling edge of STALL/STROBE –input causes the core to sample data bus. RST_X and STALL/STROBE –inputs can be driven asynchronously. Synchronizing circuitry adds some delay to signals as can be seen from timings constraints. Boot address and RST_X –signal can be driven simultaneously. A simple reset scheme might go as follows: Drive RST_X low and STALL/STROBE high while simultaneously driving boot address to data bus. Hold RST_X and boot address for a minimum of 6 clock cycles. In halfway of RST_X pulse pull STALL/STROBE low.

All delays (TDx) are relative to previous rising edge of the CLK –signal.

Table 4.1, Mnemonics and keywords in figures T.1 through T.14

keyword	**explanation**	**notes**
Propagation delays		
TD1	Delay from rising clock edge to the moment when data is valid on output of a D flip-flop.	Technology dependent.
TD2	Maximum delay of **d_cache_miss** –signal. Refer to preliminary synthesis results.	Suitable synthesis constraints for bus_req –input must be set if core is synthesised separately.
TD3	Maximum delay of the bus_req –signal. Refer to results of preliminary synthesis.	
TDZ	TD1 + delay of a tri-state gate	See synthesis notes about handling tri-state control signals.
Pulse lengths		
T6	Minimum length of interrupt pulse, synchronous mode: 2 * (TH + TS)	should be driven synchronously.
T7	Minimum length of interrupt/cop exception pulse: 1,5 * length of clock cycle.	Can be asynchronous.
T8	Minimum length of reset pulse: 1,5 * length of clock cycle(boot from zero). 6 * length of clock cycle(external boot address).	
T9	Minimum T9: 3 * length of clock cycle.	
T10	Minimum T10: 3 * length of clock cycle.	

Other time constraints		
TS	Setup time of a flip-flop: data input must have settled time TS before the next rising clock edge.	Technology dependent.
TH	Hold time of a flip-flop: data input must not change before time TH after rising clock edge.	
keywords used in diagrams		
IDLE DATA	Data and address which are driven to bus by core when bus is in idle state. Should be the values from previous access unless the bus is floated during the last RESERVED – cycle.	
IDLE ADDR		
SAMPLED DATA		
SAMPLED ADDR		
ADDR	Valid address or data of an active access.	
DATA		
IDLE	Idle cycle of the bus, no active accesses. Core drives last values on bus.	
ACCESSED	Core owns the bus and is performing write or read access.	
RESERVED	An external device owns the bus.	